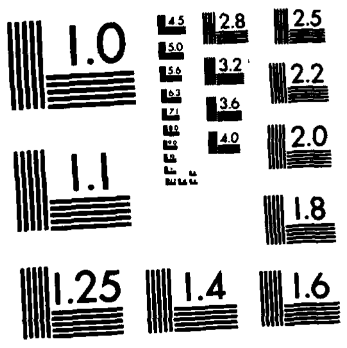


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

GENERATING AN OUT-THE-WINDOW COCKPIT IMAGE WITH THE iAPX 432

THESIS

AFIT/GCS/EE/82-12

Roger A. Cooper
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/EE/82-12

GENERATING AN OUT-THE-WINDOW COCKPIT IMAGE WITH THE iAPX 432

THESIS

AFIT/GCS/EE/82-12 Roger A. Cooper
Capt USAF

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/82-12 — 2. GOVT ACCESSION NO. A124852	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) Generating an Out-The-Window Cockpit Image with the iAPX 432	5. TYPE OF REPORT & PERIOD COVERED Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Cooper, Roger alan Capt USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology Wright Patterson Air Force Base , Ohio	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Institute of Technology Wright Patterson Air Force Base ohio	12. REPORT DATE Dec 1982
	13. NUMBER OF PAGES 132
14. MONITORING AGENCY NAME & ADDRESS(*If different from Controlling Office*)	15. SECURITY CLASS. (*of this report*) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17.

[illegible]

LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

4 JAN 1983

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

Small Computers
Simulators

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

This thesis investigation contains three areas of study. The first investigation is to study the object-oriented architecture of the Intel iAPX 432. The second investigation is to examine the requirements of the simulator and produce design and implementation details. The third investigation is aimed at studying the F111 DIG ground terrain data base and produce a new data base that the small simulator could use.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

AFIT/GCS/EE/82-12

GENERATING AN OUT-THE-WINDOW COCKPIT IMAGE WITH THE iAPX 432

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institue of Technology

Air University

in Partial Fulfillment of the

Requirements of the Degree of

Master of Science

by

Roger A. Cooper, B.S.

Capt USAF

Graduate Computer Science

December 1982

Preface

The concept of a Performance And Communications Research And Technology (PACRAT) system was discussed by Donald Knight in his December 1981 thesis. I became interested in PACRAT because of its involvement with the Intel iAPX 432 Micromainframe. I had heard about the 432 in my architecture class and decided to do my thesis using the new computer. The main intent of this thesis was to study the Intel 432 and determine if it could meet the requirements set forth by Knight in his thesis. Unfortunately all the parts for the system did not arrive and only the design and coding was done and no 432 usage was done. Hopefully a class project or follow on thesis will show how powerful the 432 is.

I would like to thank Mr. Richard McKinley of AFAMRL's Biological Acoustic Branch for his support in this project as the sponsor and hope that my work has aided him in his PACRAT research. I would also like to thank my thesis committee, Dr. Hartrum, Dr. Lamont and Capt Black, who helped guide me through this effort. Finally I would like to thank my dear wife Vicki for all her help and moral support in this effort.

GENERATING AN OUT-THE-WINDOW COCKPIT IMAGE WITH THE 432

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

AFIT/GCS/EE/82D-12

ABSTRACT

The purpose of this thesis investigation is to determine if a small computer can generate graphic images in a real time environment. The specific computer to be used is the Intel iAPX 432 Micromainframe. The graphic images are to be used as part of an aircraft simulator in a communication performance evaluation test.

This thesis investigation contains three areas of study. The first investigation is to study the object-oriented architecture of the Intel iAPX 432. The second investigation is to examine the requirements of the simulator and produce design and implementation details. The third investigation is aimed at studying the F111 DIG ground terrain data base and producing a new data base that the small simulator could use.

The results of the three areas of investigation differ in their achievements. The study of the Intel iAPX 432 was completed through reading manuals and compiling code. The simulator design and implementation was accomplished through use of structured design techniques and the ADA programming language. The data base was changed from machine internal format to a human readable format but a usable data base was not produced.

CHAPTER 1

INTRODUCTION

BACKGROUND

Small computer processing power has been increasing since small computers first appeared on the market. However, generating solid three dimensional figures for a simulator requires a large amount of processing power (Ref 23:25). Historically, simulators have used large mainframes or special purpose computers to handle the processing power required. If the requirements for the simulator were reduced to the minimal amount of throughput necessary to continue the human interface, then the processing power requirements would be reduced. Can a minicomputer meet the processing power requirements of a minimal simulator? To answer this question the Biological Acoustics Branch of the Air Force Aerospace Medical Research Laboratory (AFAMRL/BBA) is researching audio communications which requires a minimal simulator to add a degree of authenticity to associated tests.

AFAMRL/BBA desires a simulator with solid three dimensional displ[illegible]s of [illegible] ground terrain and of other test subject airplanes th[illegible] are in view. The PACRAT, Performance

And Communication Research And Technology, system is the test which requires the use of the simulator. PACRAT is a system in which subjects' audio communication skills are tested. The main skill tested is hearing with interference from outside sources. The current test (Ref 26) includes trying to maintain a crosshair inside a moving circle on a video display while the audio communications are going on. The simulator is to replace the crosshair in the moving circle with a more realistic situation. The current test results include how often the subject kept the crosshair inside the circle and how frequently the test subject understood the commands given by the test conductor over the audio communication lines. The preliminary requirements (Ref 23) of the PACRAT system are summarize in Chapter 2 of this thesis.

After much studying and compiling of facts (Ref 23) it became evident that a digitized data base is required to prodûce the ground terrain picture. Reference 23 described the available data bases and concluded that the F111 DIG digitized ground terrain data base was best suited to the PACRAT system needs. This data base has a 32 bit wide key to access the data base. The 32 bit wide key of the data base produces a requirement for a 32 bit computer. There are several new 32 bit small computers available on the market today. After reviewing the literature on these new computers, Reference 23 concluded that the Intel iAPX 432 might provide the processing power required to provide a minimal simulator.

SCOPE

This thesis investigation will study the use of the Intel iAPX 432 Micromainframe, its object oriented architecture, and the ADA programming language to support the PACRAT system. ADA is the only programming language supported on the iAPX 432. Programs necessary for the PACRAT system will be developed through top down design techniques. The development will consist of the following:

-- Requirements review

-- Produce high level design of the system

-- Produce low level functional design of the modules in the system

-- Implement the design in ADA

-- Test and integrate the system

The final area of study will consist of examining the F111 DIG data base. This data base is needed to produce the ground terrain picture. This data base is currently on a tape with a Singer-Link operating system file format (Ref 26). This part of the thesis effort will attempt to read the tape and convert the internal machine format data into human readable format and then into a useable data base for the PACRAT system.

CURRENT KNOWLEDGE

This thesis investigation will continue from Reference 23. The following information was obtained from Reference 23.

-Requirements as described by the sponsor which will be summarized in Chapter 2.

-SADT charts for displaying the requirements which will be used in Chapter 2.

-Hidden line algorithm for drawing screen images as seen by the author which will be used in Chapter 4.

The sponsor has a completed FORTRAN program for calculating the new position of a C135 cargo aircraft. This program will be used in the simulator to determine the new position.

The following are still needed to implement the system.

-A ground terrain data base.

-A program design of the system flow.

-Timing studies of the iAPX 432 performing the task.

-A system integrating all the parts.

APPROACH

There will be three major efforts in this effort aimed at implementing the PACRAT system. The first effort will be to try to acquire the F-111 visual data base for PACRAT. After acquiring the data base then work the data base into a useable form for PACRAT. The second effort will be to analyze the Intel iAPX 432 architecture in relation to the PACRAT requirements. ADA will be studied for use in the 432 environment. The third major effort of this thesis investigation will analyze the requirements set forth (Ref 23:118) and produce an implementation for testing. The implementation is to be a design of an Out-The-Window Cockpit Image generator. The design will deal with reviewing the requirements already established (Ref 23) and the sponsor's new requirements (Ref 26). The Structured Analysis and Design Technique, SADT (Ref 39), approach will be studied and transformed into Data Flow Diagrams, DFDs (Ref 39), to show how requirements transform into design. Structure charts (Ref 39) will be presented to display the heirarchy of the packages and procedures in the system. Then the implementation of the programs will be achieved. The flow of data between modules, how multi-tasking will be used, and the timing requirements will be presented. Finally the results of the design work and the timing results of the program on the Intel iAPX 432 will be compared with the requirements stated in Chapter 2.

SEQUENCE OF PRESENTATION

Chapter 2 will deal with the presentation of the requirements already established (Ref 23) and the sponsor's new requirements (Ref 26). The Structured Analysis and Design Technique, SADT (Ref 39), approach will be used to show the requirements of the system.

Chapter 3 will deal with the presentation of the high level design of the system. Data Flow Diagrams (Ref 39) will be used to show the design of the system and flow of the data.

Chapter 4 will deal with the implementation of the design. Structure charts will be used to show the low level implementation. The flow of data between modules, how the multi-tasking will interface, and the timing requirements analysis will be presented.

Chapter 5 will deal with the results of the implementation work. The results will be compared against the criteria set forth in Chapter 2 and against the use of the Intel 432 for the Air Force.

Chapter 6 will deal with conclusions and recommendations. The conclusions reached and the recommendations for follow-on work will be presented.

Appendix A will deal with reviewing the Intel iAPX 432, its architecture, the ADA programming language and how they fit together to produce a PACRAT work station.

CHAPTER 2

REQUIREMENTS

INTRODUCTION

The purpose of this chapter is to define, restate, and update the requirements for the PACRAT system as currently envisioned by the sponsor through the use of the Intel iAPX 432 micromainframe computer. First, a table is presented reviewing the results of previous work. Second, updated requirements and further explanation of requirements are presented. Finally, the requirements are summarized.

THE PACRAT SYSTEM

The PACRAT system consists of one test conductor in audio communication with ten test subjects. The communications between them is accomplished by commands from the test scenario. The reactions of the test subjects will be recorded on disk. The record will include responses to the audio commands in the form of flight parameter information from the simulator.

Each test subject will be at a dedicated test station. A test sation will include a disk drive, a video display, a keyboard, a throttle, a joystick, a microphone, 1024 switches, a lightpen, a Z80 computer to read the devices and the iAPX 432 to do the computations necessary to maintain the video display as an out-the-window cockpit image. The scenario envisioned using the simulator include landing and taking-off, landing and taking-off under attack from other test subjects, flying in a dog-fight, and flying a straight and level glide path. The general flow of data begin with the test conductor reading the test scenario and issueing commands to the test subjects. The test subjects hear the commands and react. The computer will record the test subjects' reactions and act on the reactions to produce a new image for each subject. To produce the new image the computer must determine which other test subjects are sightable, retrieve from the data base the descriptions of the sightable aircraft, determine the new location, retrieve from the data base the description of the ground terrain, and from all of these descriptions generate a new out-the-window cockpit image.

Figure 1 and 2 show the overall requirements of the system. Figure 3 and Figure 4 show the requirements of the next level of the system. Table I shows the correlation between the node designator and the figure numbers and it includes the title of the figure.

NODE	TITLE	FIGURE
A-0	PACRAT System	1
A0	Test System	2
A1	Test Conductor	3
A2	Test Subject	4

Table I Correlation between Nodes and Figures

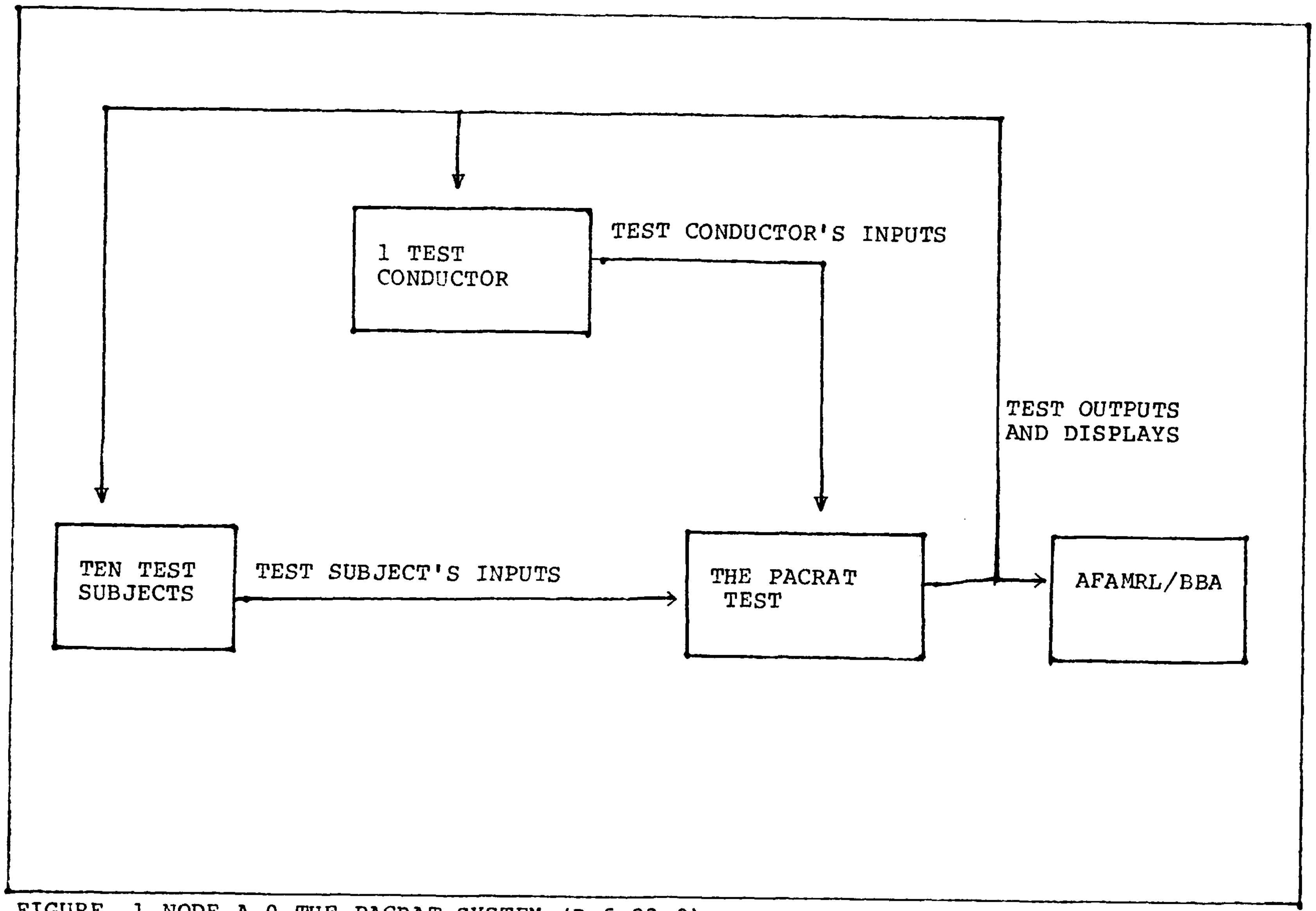


FIGURE 1 NODE A 0 THE PACRAT SYSTEM (Ref 23:9)

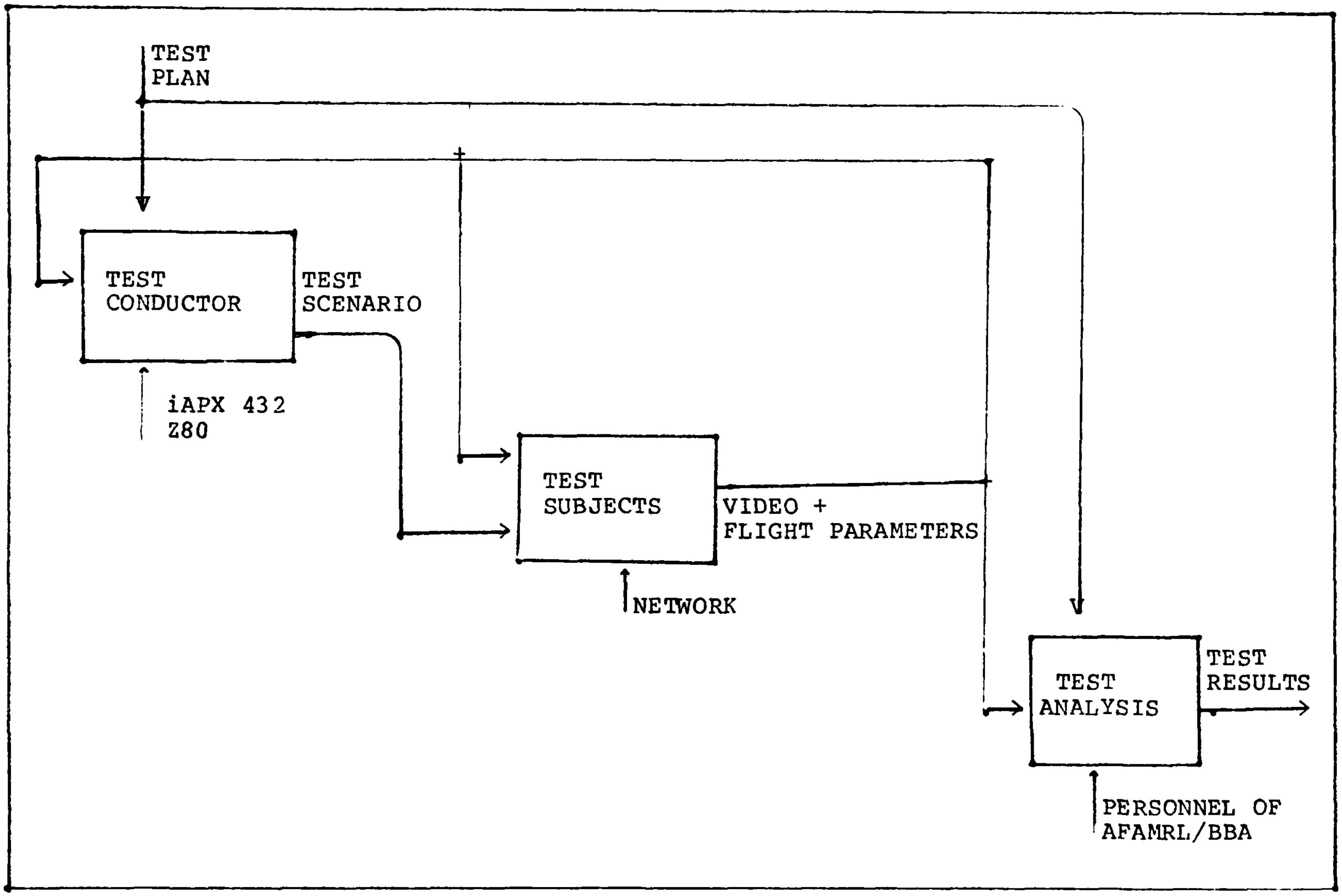


FIGURE 2 NODE A0 THE PACRAT TEST (Ref 23:11)

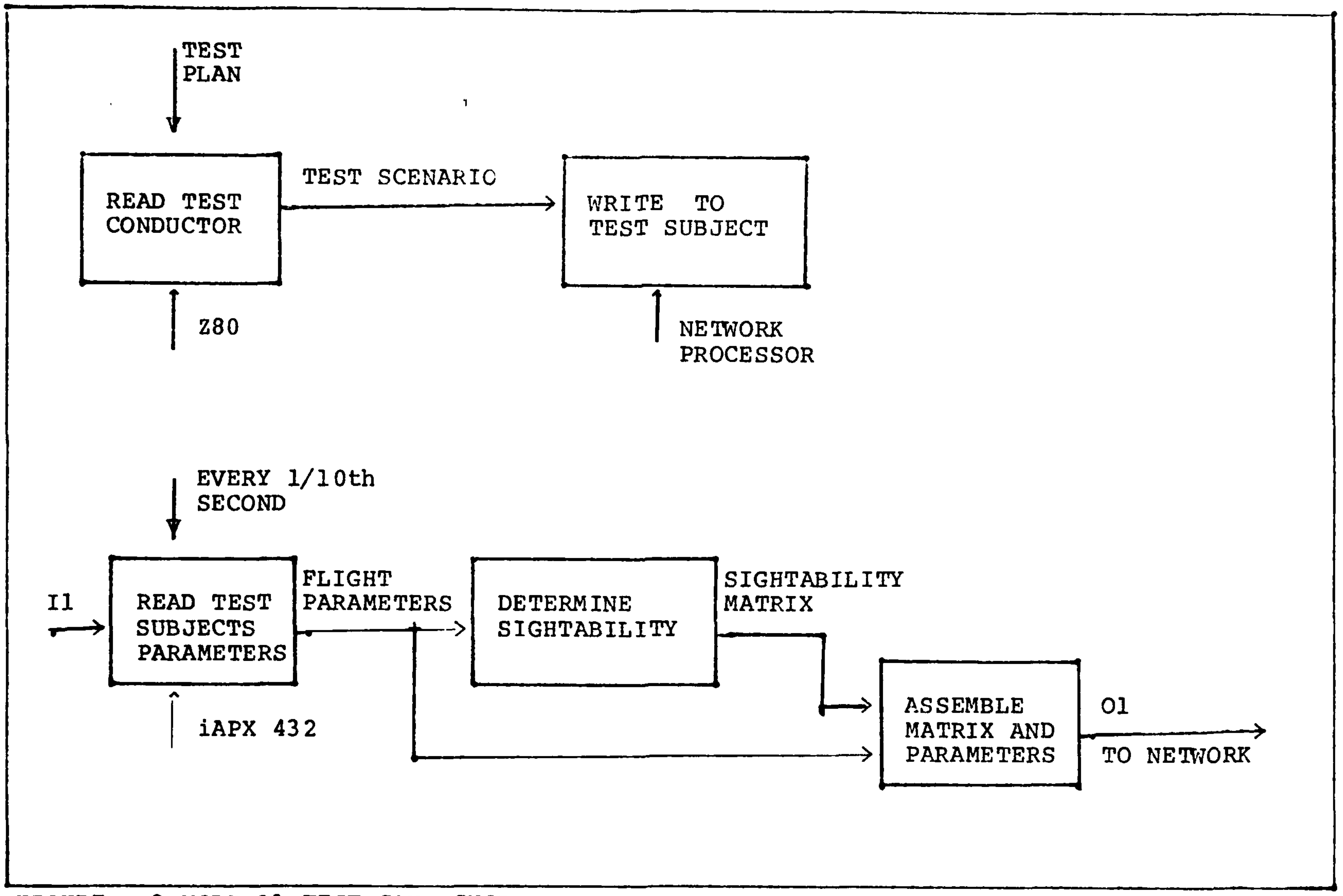


FIGURE 3 NODE A1 TEST CONDUCTOR

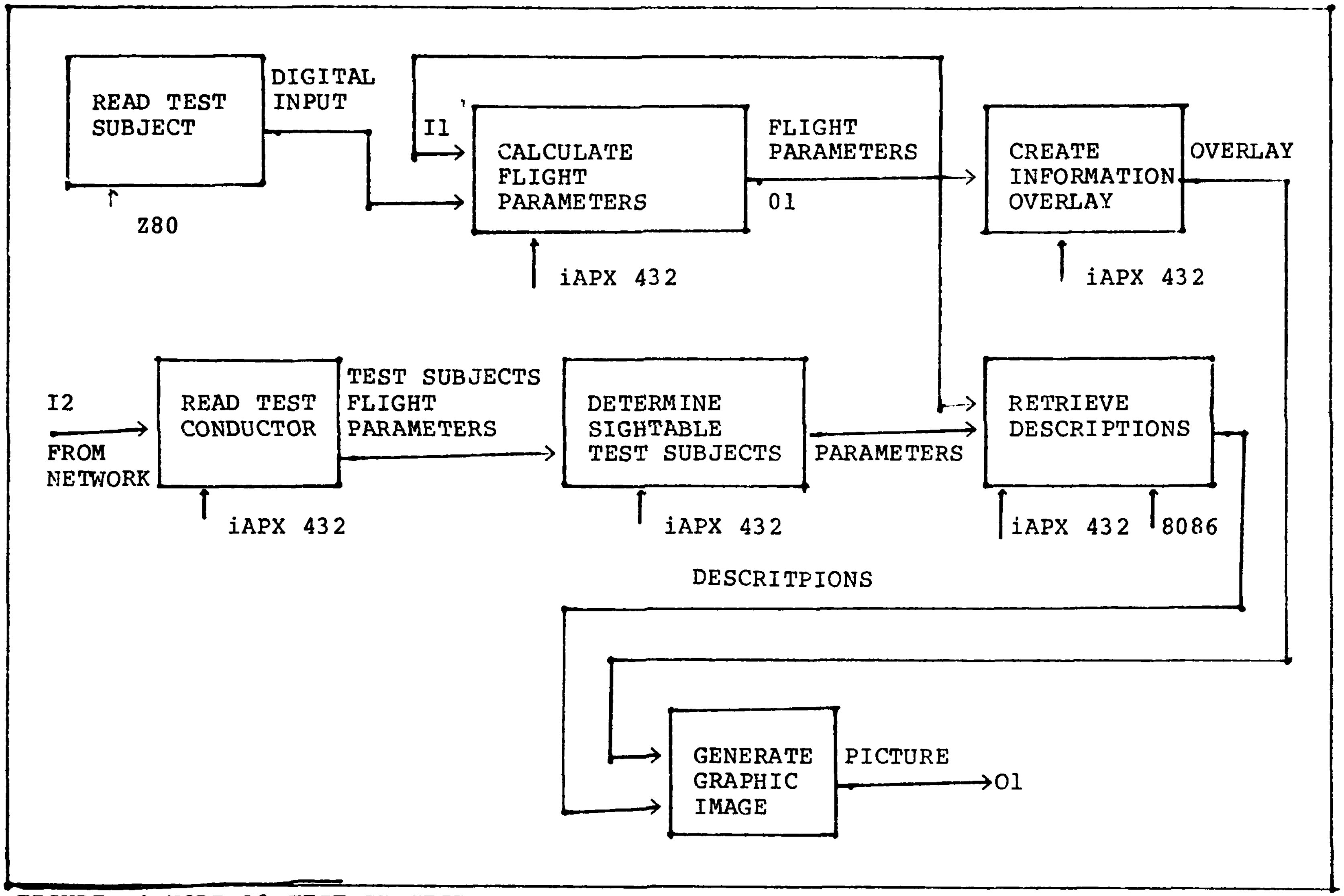


FIGURE 4 NODE A2 TEST SUBJECT

REVIEWED REQUIREMENTS

To understand the PACRAT system it was necessary to understand the background and the requirements. The background was presented in Chapter 1 and the table below summarizes the requirements. Table II was generated by study (Ref 23) and through discussions with the sponsor(Ref 26). The table was broken into two parts. The first part, general requirements, refers to the requirements of the system. The second part, specific requirements, refers to the requirements of the out-the-window image generator.

TABLE II TABLE OF REQUIREMENTS

0.REQUIREMENTS	1.0 GENERAL REQUIREMENTS 2.0 SPECIFIC REQUIREMENTS
1.GENERAL	1.1 PERSONNEL 1.2 ENVIRONMENT 1.3 COST 1.4 SYSTEM CONSTRAINTS 1.5 RELIABILITY 1.6 SYSTEM INITIALIZATION 1.7 TEST
1.1 PERSONNEL	1 TEST CONDUCTOR 10 TEST SUBJECTS
1.2 ENVIRONMENT	2 ROOMS
ROOM 1	19 X 27 FEET ACOUSTIC CHAMBER HOLDS TEST SUBJECTS AND EQUIPMENT
ROOM 2	HOLDS TEST CONDUCTOR
1.3 COST	$400,000 ACQUISTION 20% OF FINAL YEAR ACQUISTION COST FOR MAINTENANCE AND SYSTEM UPGRADE
1.4 SYSTEM CONSTRAINTS	Z80 MICROPROCESSOR AS INPUT/OUTPUT CONTROLLER INTEL 432 AS TEST

	SUBJECTS GRAPHICS PROCESSOR
1.5 RELIABILITY	ONE TEST SUBJECT ERROR SHOULD NOT AFFECT ANY OTHER TEST SUBJECT
MIMIMAL OPERATING TIME	4 HOURS A DAY/ 5 DAYS A WEEK
MTBF	NLT 3 DAYS
MTTR	NMT 1 HOUR
1.6 SYSTEM INITIALIZATION	NON-REAL-TIME FUNCTION
POWER UP	Z80 DIAGONISTIC SOFTWARE CHECK OUT SYSTEM
CONFIGURE NETWORK	Z80 REPORTS TO TEST CONDUCTOR CONSOLE
UPLOAD PROGRAMS	TEST CONDUCTOR WILL DICTATE WHICH PROGRAM IS TO BE LOADED FOR THE TEST
PERSONNEL	1 MAN
TIME	1 HOUR
1.7 TEST	1.7.0 REAL-TIME FUNCTION 1.7.1 TIME FRAME 1.7.2 READ TEST SUBJECTS INPUTS 1.7.3 READ TEST CONDUCTORS INPUTS 1.7.4 RECORD TEST DATA 1.7.5 UPDATE TEST OUTPUT AND DISLAYS 1.7.6 POSITIONAL PARAMETERS 1.7.7 TEST CONDUCTORS DISPLAY 1.7.8 TEST SUBJECTS DISPLAY
1.7.1 TIME FRAME	10 TO 7 TIMES A SECOND
1.7.2 READ TEST SUBJECTS INPUTS	
ANALOG DEVICE	THROTTLE AND JOYSTICK
AUDIO DEVICE	MICROPHONE,WHEN STARTED OR STOPPED SEND A SIGNAL TO THE Z80
DIGITAL DEVICE	LIGHTPEN, ALPHANUMERIC KEYBOARD, AND 1024 SWITCHES
1.7.3 READ TEST CONDUCTOR INPUTS	
ANALOG DEVICE	TRACKBALL,THROTTLE AND JOYSTICK
AUDIO DEVICE	TAPE RECORDER MICROPHONE, WHEN STARTED OR STOPPED

DIGITAL DEVICE	SEND A SIGNAL TO THE Z80 LIGHTPEN, ALPHANUMERIC KEYBOARD, AND 1024 SWITCHES

1.7.4 RECORDING TEST DATA

WHEN	TERMINATION OF AUDIO SIGNAL OR INTERRUPT FROM DIGITAL INPUT DEVICE
STORAGE	1500 RECORDS
WHAT	
INTELLIGIBILITY TEST	TIME, STATION, AND CORRECT WORD RECORDED
AERIAL MAP FOLLOWING	TIME, STATION, POSITION, AND LENGTH OF TRANMISSION

1.7.5 UPDATING TEST OUTPUT AND DISPLAYS

DIGITAL INPUTS	VISUAL INDICATOR OF STATUS OF SWITCH
AUDIO INPUTS	SPEECH SYNTHESIZER GENERATION WHEN REQUIRED
CRTS	COLOR,740 X 520 PIXEL RESOLUTION
DISPLAYS	ALPHANUMERIC TEXT
	TWO DIMENISION STICK FIGURES WITH TEXT
	THREE DIMENISION STICK FIGURES WITH TEXT WITHOUT HIDDEN LINE REMOVAL
	THREE DIMENISION STICK FIGURES WITH TEXT WITH HIDDEN LINE REMOVAL
	SOLID THREE DIMENISION FIGURES WITH TEXT

1.7.6 POSITION PARAMETERS

MAXIMUM VELOCITY	1.5 MACH
MAXIMUM ROLL	360 DEGREES PER SECOND
MAXIMUM PITCH	150 DEGREES PER SECOND
MAXIMUM YAW	150 DEGREES PER SECOND
WIND LOFT	0
FIELD OF VIEW	
HORIZONTAL	120 DEGREES
VERTICAL	+60 AND -30 DEGREES
MAXIMUM VISIBILITY	10 MILES
MINIMUM DISTANCE	15 FEET
GAMING AREA	600 MILE SQUARE
	ABILITY TO RECIEVE 9 OTHER AIRCRAFT POSITIONS

1.7.7 TEST CONDUCTOR DISPLAY	12 CRTS
1 CRT	SCRIPT TEXT
10 CRTS	SAME VIEW AS EACH TEST SUBJECTS
1 CRT	BIRDS EYE VIEW WITH ZOOMING IN POSSIBLE

1.7.8 TEST SUBJECTS DISPLAY	1 CRT
	GROUND,TEXT, AND OTHER TEST SUBJECTS

---

2. SPECIFIC	2.1 CALCULATE AIRCRAFT PARAMETERS
	2.2 GAMING DATA BASE
	2.3 ACTIVE DATA BASE
	2.4 OUTPUT FRAME
	2.5 TYPE OF OPERATIONS
	2.6 NETWORK

2.1 CALCULATE AIRCRAFT	PARAMETERS
JOYSTICK ACCURACY	2.81 DEGREES
THROTTLE ACCURACY	.78
TIME	100 TO 141 MILLISECONDS
BANDWIDTH BETWEEN Z80 AND INITIAL PROCESSOR	30 BYTES PER SECOND
PERSPECTIVE	3X1X32 AIRCRAFT IN WORLD COORDINATE SYSTEM
VIEW UP	3X1X16 DIRECTION OF YAW
VIEW PLANE NORMAL	3X1X16 DIRECTION OF ROLL
GROUND SPEED	16 BITS KNOTS
FUEL REMAINING	16 BITS POUNDS

2.2 GAMING DATA BASE	
SIZE COVERED	600 MILE SQUARE, BY 3/8 OF INCH INCREMENTS
SIZE OF STORAGE	7.68M BYTES
SIZE OF KEY	32 BITS
FROM	F111 DIG

2.3 ACTIVE DATA BASE	20K X 32 BITS -- HAS 20 SQUARE MILES OF GROUNG TERRAIN PLUS OTHER PLANES IN VIEW

2.4 TYPES OF MATH OPERATIONS	INTEGER

2.5 OUTPUT FRAME	740 X 520 X 8 BITS
OTHER OUTPUT FRAME	COMPUTER BY INITIAL GRAPHICS PROCESSOR AS OVERLAY

FLIGHT DATA	MAGNETIC HEADING ALTITUDE AIRSPEED
FLIGHT INFORMATION	TEXT
2.6 NETWORK	
CONFIGURATION	STAR
NETWORK PROCESSOR TO DATA BASE	232 BYTES PER SECOND
DATA BASE SIZE	327K BYTES
CAPACITY OF LINES	204K BYTES PER SECOND ACCORDING TO KNIGHT (Ref 23:69) 3170 BYTES PER SECOND

TABLE II TABLE OF REQUIREMENTS

The table above presented facts only, with no discussion. The interested reader should read Reference 23 for a discussion of the requirements or Reference 26 for updated requirements.

CHANGES FROM PREVIOUS REQUIREMENTS

The following are changes or corrections to the requirements set forth in Reference 23 through study or discussion with the sponsor.

1. The sponsor changed the number of test subjects from nine to ten because of greater space available in the room where the PACRAT system is to be located. The location parameter data block contains the information of all the aircraft. If the aircraft is within viewing area, then the data necessary to display the aircraft is available in this data block. Since the number of test subjects changed, the

format of the location parameter data block from the test conductor to the test subject changed. The location parameter data block has 10 instead of 9 aircraft locations and its format is as follows:

Bit-Bit	Description
0000-0031	Header
0032-0127	Perspective of Aircraft 1
0128-0175	View-Up of Aircraft 1
0176-0223	View-Plane-Normal of Aircraft 1
0224-0419	Aircraft 2
. .	.
. .	.
1988-2184	Aircraft 10

2. The sponsor changed the number of test subject input switches from 256 to 1024 because one aircraft was designed with over 400 switches and the new configuration of the Z80 allowed for 1024.

3. Each test subject will have a separate disk drive because of availability of cheaper disk drives. Therefore the original timing and bandwidth requirements are no longer valid because timing was based on sharing of a disk. The disk was going to be shared due to the high cost of disk drives when the original plan was devised.

4. The audio record format was updated to correct a counting error found in the original requirements.(Ref 23:70). An audio record is written to disk each time an audio transmission is made.

Bit-Bit	Description
000-003	Console Number
004-031	Time
032-047	Start Time
048-063	End Time
064-159	Perspective
160-207	View-Up
208-255	View-Plane-Normal

5. The amount of storage required at each test subject's station for the audio records is 46080 bytes. This can be accomplished because in change 3 above, disks are available at each test subject's station. The amount of storage is different because of the counting error found in change 4 above.

The average time between audio signals is 2.5 seconds. The average time of a test session is 1 hour. The number of records needed to be stored is 1440.

```
1440 records/session := 1 record / 2.5 seconds *
                              3660 seconds / session;
```

Each record according to change 4 is 256 bits long or 32 bytes long. The amount of storage necessary per session is 46080 bytes.

```
        46080 bytes / session := 32 bytes / record *
                                 1440 records / session;
```

The amount of data that could be off loaded from the network would be 10 times the amount of storage needed per user since there are 10 users.

```
       460800 bytes / test    := 46080 bytes / session *
                      10 test subject sessions / test;
```

6. The amount of data transmitted over the network has been changed. Change 1 above caused the amount of data from test conductor to test subject to increase because of the location parameter data block increase. Change 5 above caused a decrease of 460800 bytes per test session. The amount of traffic stated in Reference 23 cannot be verified so new calculations are necessary. The new calculation is as follows:

The messages over the network fall into the following catagories:

1. Location parameter data block from test subject to test conductor is 28 bytes long. Its format is the first 28 bytes of the location parameter data block from the test conductor to test subject. This data is transmitted 10 times every second.

2. Location parameter data block from test conductor to test subject is 273 bytes long. Its format is layed out in change 1 above. This data is transmitted 10 times every second.

3. The switch status data block from test subject to test conductor is 8 bytes long. Its format is a 32 bit header followed by 32 bits for switch number and status of switch. This data is transmitted at its worst case 10 times every second.

4. The audio signals from the test conductor to the test subject is 32 bytes long. Its format is described in change 4 above. This data is transmitted every 2.5 seconds.

```
  28 bytes / record * 10 records / second + -- catagory 1
 273 bytes / record * 10 records / second + -- catagory 2
   8 bytes / record * 10 records / second + -- catagory 3
  32 bytes / record * record / 2.5 seconds  -- catagory 4
  := 3170 bytes / second := 25360 bits / second;
```

7. The timing constraints were caused by the number of calculations and the access time to the disk drive. The number of calculations are undefined because they depend on the data format and the algorithms used. The data format used in Reference 23 leaned toward Pascal data types and the implementation is being done in ADA. ADA has more

appropriate data types and an algorithm with less computations can be used.

8. Reference 23 looked at 4 different computers. This thesis will look at only one. The Intel iAPX 432 will be evaluated as the graphics processor and the initial graphics processor because it is the must promising processor evaluated to date (Ref 23). The number of processors will be determined to meet the requirements (Ref 23:103).

SUMMARY

Except for the differences stated above, this author has evaluated and agrees with prior work (Ref 23). Using Reference 23 as a starting point and the Intel iAPX 432 computer for direction, the requirements for the PACRAT system have been presented. Most of the requirements stated above are for the PACRAT system and not directly related to the generation of graphics display which is what this thesis is examining. The main thrust of this thesis effort is to examine the Intel iAPX 432 microcomputer using the ADA programming language while multi-proramming tasks, and determine if the Intel iAPX 432 meets the requirements set forth. ADA will be used because it is the only language available for the iAPX 432.

CHAPTER 3

DESIGN

INTRODUCTION

The purpose of this chapter is to present PACRAT's high level design and its test plan. The previous chapter dealt with the system requirements. This chapter will detail how PACRAT is structured, how the programs are to be tested and the rational leading to design decisions.

The design is of an aerial map-following type of test with the out-the-window cockpit image being solid three dimensional images as described in chapter 2 requirement 1.7.5. This structure was chosen because it has the most computations per time frame. First the conductor's station will be discussed followed by the test subjects' station. The conductor's station major tasks, as described by figure 3 in chapter 2, are information gathering and information dispersal. The test subjects's station major tasks, as described by figure 4 in chapter 2, are initialization, determining the location of the test subject, determining what other test subjects are visible, and generating the graphics display.

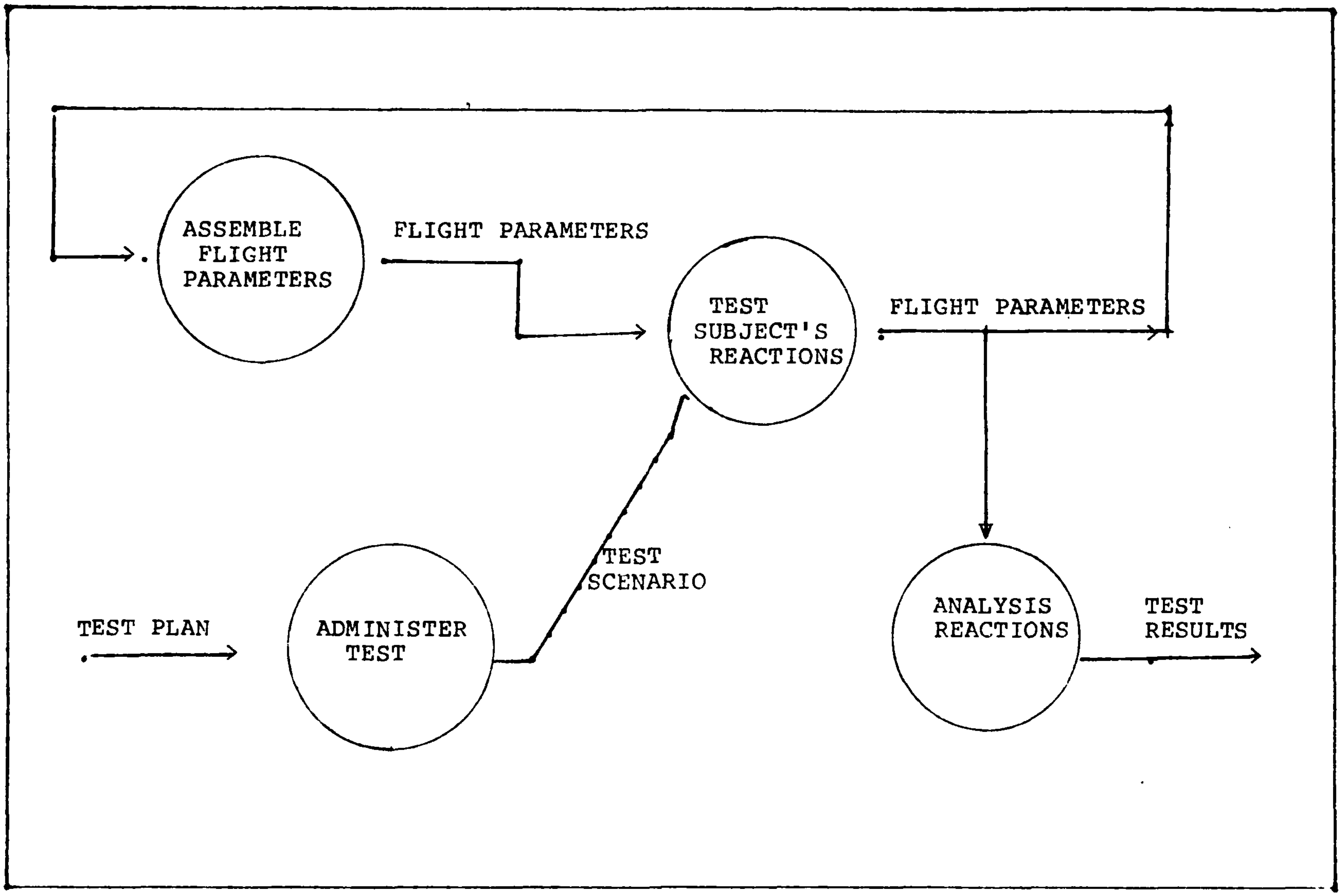


FIGURE 5 DIAGRAM 0 THE PACRAT SYSTEM

The test plan will discuss how each function will be tested. The plan will include how the test subject's station will be tested, how the test conductors's station will be tested and will explain how the entire system can be tested.

Data flow diagrams, DFDs, are used to show the flow of data through the system. With the information displayed, it can be determined what processes can be done concurrently.

Table III shows the correlation between the SADT charts of the requirements and the DFDs of the design. Figures 5,6 and 8 are DFDs produced from the requirements. Figure 7 is an expansion of bubble number 2 in Figure 6.

Figure Number	DFD Diagram Number	SADT Node NUmber	Figure Number	Name
5	0	A0	1	The PACRAT System
6	1.0	A1	2	The Test Conductor
8	2.0	A2	3	The Test Subject

Table III Correlation between SADT and DFDs

CONDUCTOR'S STATION

Since the network of the test subjects is in a star configuration, as specified in requirement 2.6 of chapter 2, the location of each test subject must be sent to the test conductor, then assembled and sent to each test subject. An ADA package will include all the descriptions of the variables to be used by the conductor and the routines to read and write all of these variables. A package is what ADA uses to put like functions in one area for easier maintenance. After discussion with the sponsor, it was decided that the calculations for the determination of which test subjects can see other test subjects will be done at the test conductor's station. This will save fifty percent of the calculations since all ten subjects will not be performing the calculations to see the other test subjects. This package reflects what is happening in bubble 1 of diagram 1.2. Diagram 1.2 will take the distance calcũlations and determine if the subjects are within the viewing pyramid according to the requirements (Ref 23). The package will be written such that if this decision is later changed this package will not be changed.

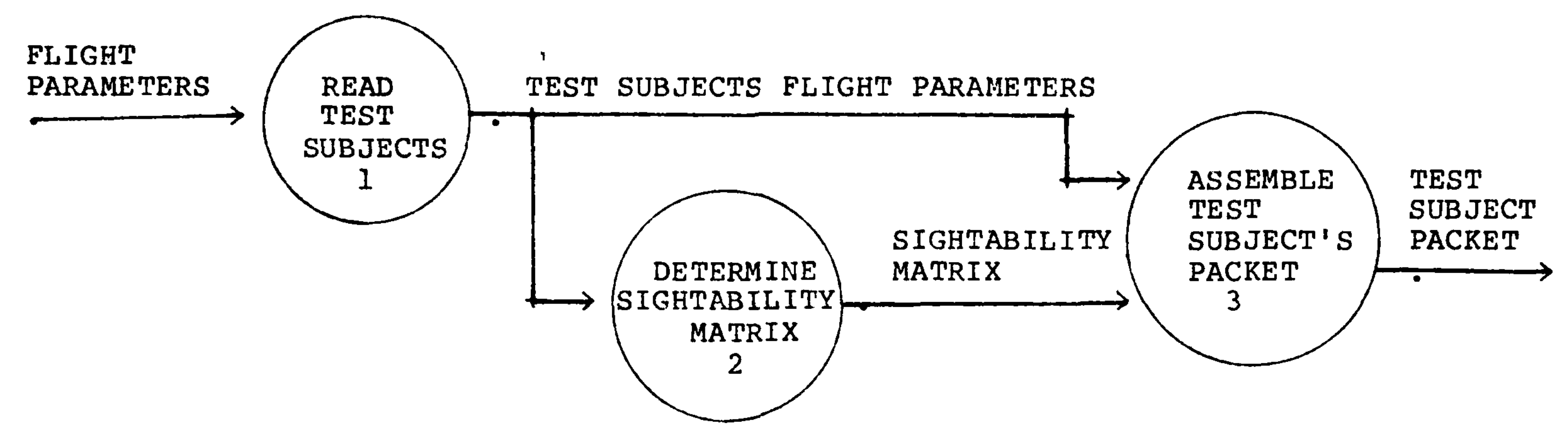

FLIGHT PARAMETERS
READ TEST SUBJECTS 1
TEST SUBJECTS FLIGHT PARAMETERS
DETERMINE SIGHTABILITY MATRIX 2
SIGHTABILITY MATRIX
ASSEMBLE TEST SUBJECT'S PACKET 3
TEST SUBJECT PACKET

FIGURE 6 DIAGRAM 1.0 PACRAT TEST CONDUCTOR

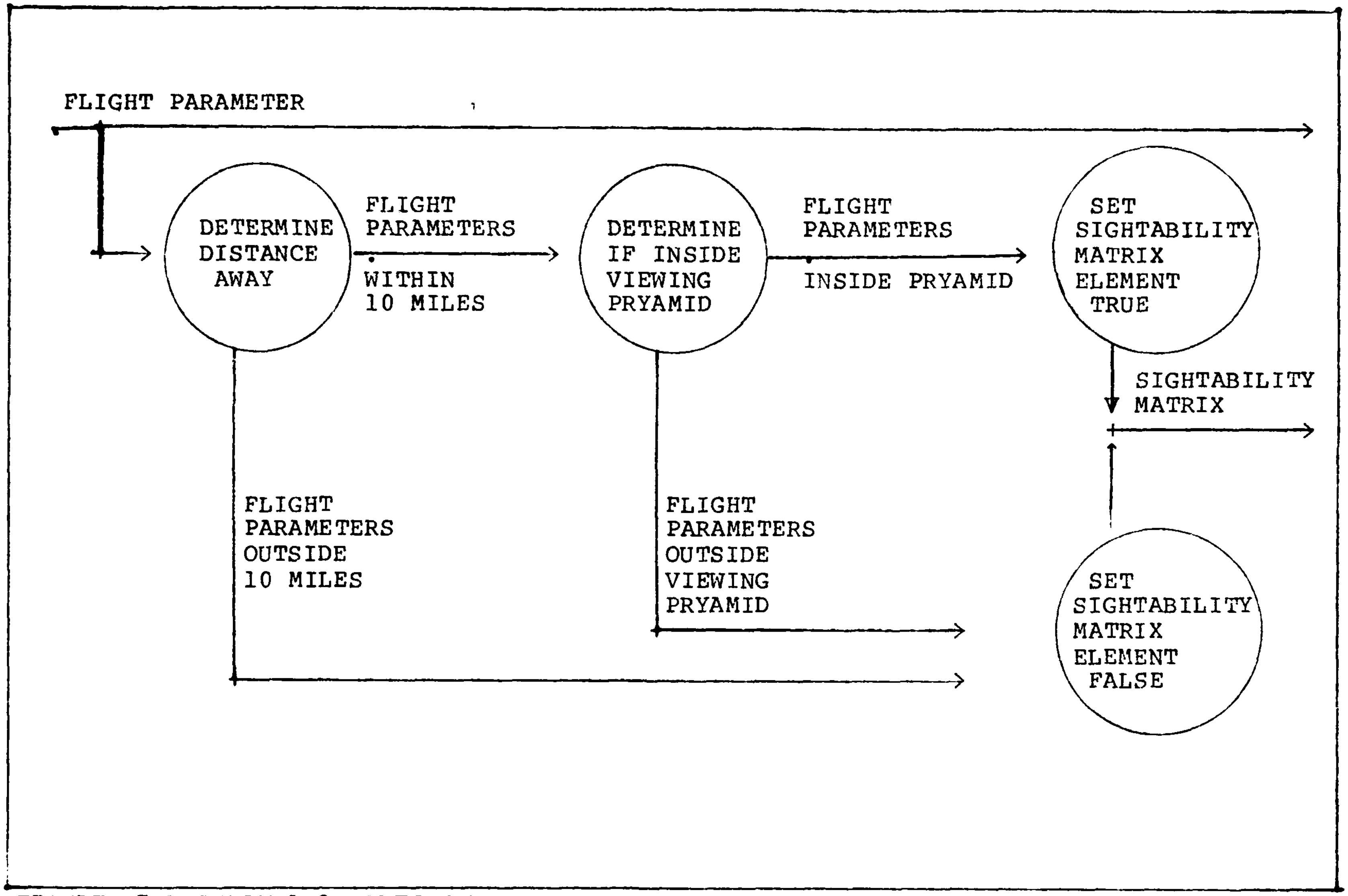

FLIGHT PARAMETER
DETERMINE DISTANCE AWAY
FLIGHT PARAMETERS WITHIN 10 MILES
DETERMINE IF INSIDE VIEWING PRYAMID
FLIGHT PARAMETERS INSIDE PRYAMID
SET SIGHTABILITY MATRIX ELEMENT TRUE
SIGHTABILITY MATRIX
FLIGHT PARAMETERS OUTSIDE 10 MILES
FLIGHT PARAMETERS OUTSIDE VIEWING PRYAMID
SET SIGHTABILITY MATRIX ELEMENT FALSE

FIGURE 7 DIAGRAM 1.2 DETERMINE SIGHTABILITY MATRIX

TEST SUBJECT'S STATION

The test subject's station must perform many functions within its 1/10th of a second time frame (Ref Requirement 1.7.1 of Chapter 2). These functions include reading-from and writing-to the test conductor, determining what the new location is, determining what other aircraft the test subject can see, retrieving the description of those aircraft, retrieving the description of the terrain around the new location, making a new picture of the area, and making an overlay of the picture with basic flight information on it. The goal is to reduce the wall clock time, therefore execute these functions at the same time.

The test subject program must find out some basic information before it can begin. This basic information is: its starting parameters, the type of aircraft, and its identification code. This will be handled by a package which asks the test subject for this basic information and returns it to the main routine. The alternative is to write a single program per station giving the program the unique id, starting parameters and only one type of aircraft. This is a very restrictive program and does not provide for the expansion of the eariler program. Either way, bubble 1 of diagram 2.0, figure 8 must be done.

Bubble 2.4 can be handled by a simple check of the boolean variable set up by the conductor's program. The function first reads the test conductor's data, sink 1 of diagram 2.0, and then checks the boolean variable array. If this boolean variable is true, then the program must retrieve the type of aircraft that the test subject is seeing and obtain the description of the aircraft from the data base, bubble 2.5.3, and pass this information and the sightable aircraft's flight parameters to make a correct graphic image, bubble 2.6.

At the same time that the determination of sightability is processing, the test subject can also be determining his new location, bubble 2.2. After determining the new location, the description of the terrain must be retrieved, bubble 2.5.1, and sent on to the graphics image processing package, bubble 2.6. Also the new location parameters must be sent to the test conductor, sink 2 of bubble 2.0, and to an overlay-creating program, bubble 2.3.

The graphics processing package can start whenever there are some descriptions to process. It must take the descriptions and remove those portions of the description that cannot be seen. Then it must shade in the surfaces with the correct color and then determine the color and intensity of the screen pixels. This has been an extremely simplified description of the processing that goes on in creating the three dimentional solid graphic image. The package, make_pictures, shown in attachment 10, will contain the detailed algorithm of the process for the interested reader.

The overlay program will accept the new location parameters and the old location parameters and create a new overlay. After determining which parameters have changed, the overlay must be changed and the pixels identified. After all the parameters have been processed and pixels changed, the overlay will be sent to the graphics display generator.

Figure 8 is the Data Flow Diagram of the SADT node A2 from Figure 4. Figures 9, 10, 11, 12, 13, and 14 are the next level down expansion of Figure 8.

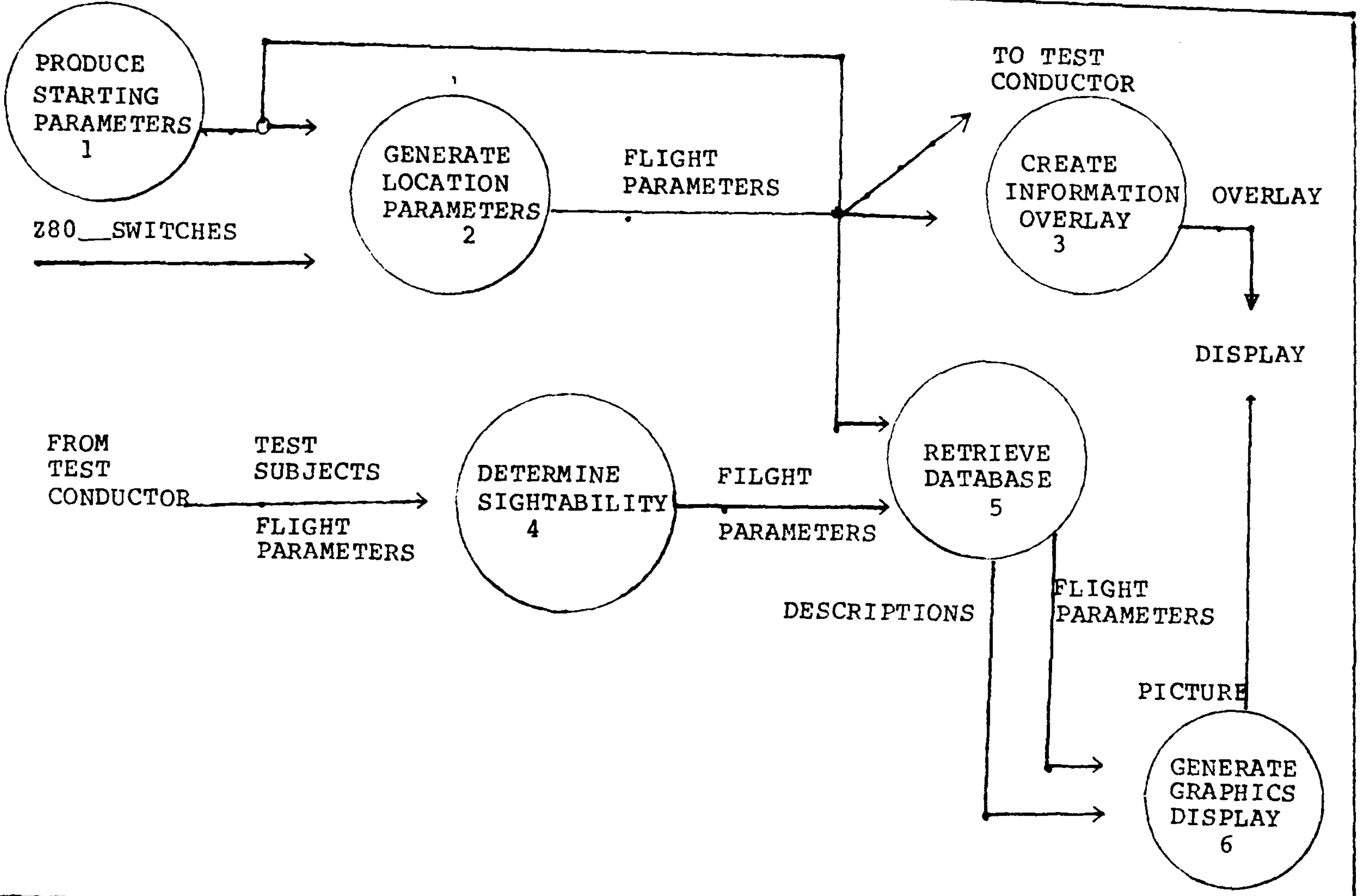


FIGURE 8 DIAGRAM 2.0 PACRAT TEST SUBJECT

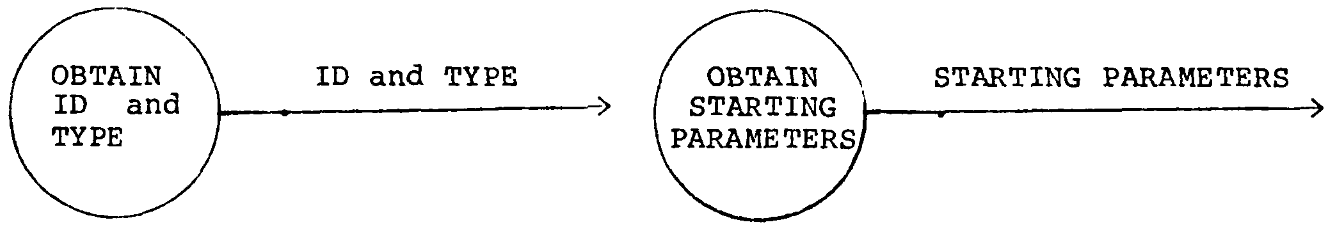


FIGURE 9 DIAGRAM 2.1 PRODUCE STARTING PARAMETERS

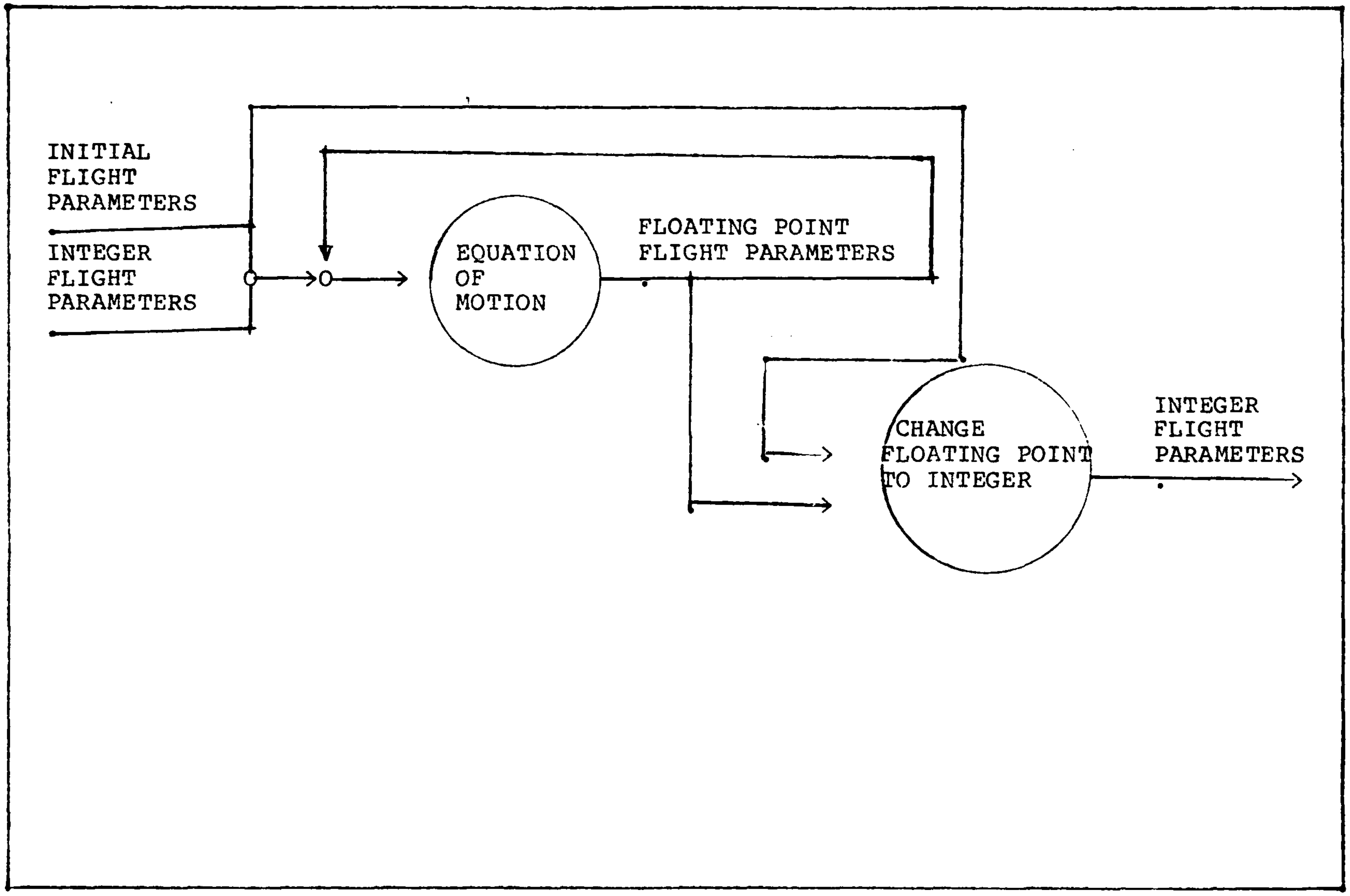

INITIAL FLIGHT PARAMETERS
INTEGER FLIGHT PARAMETERS
EQUATION OF MOTION
FLOATING POINT FLIGHT PARAMETERS
CHANGE FLOATING POINT TO INTEGER
INTEGER FLIGHT PARAMETERS

FIGURE 10 DIAGRAM 2.2 GENERATE LOCATION PARAMETER

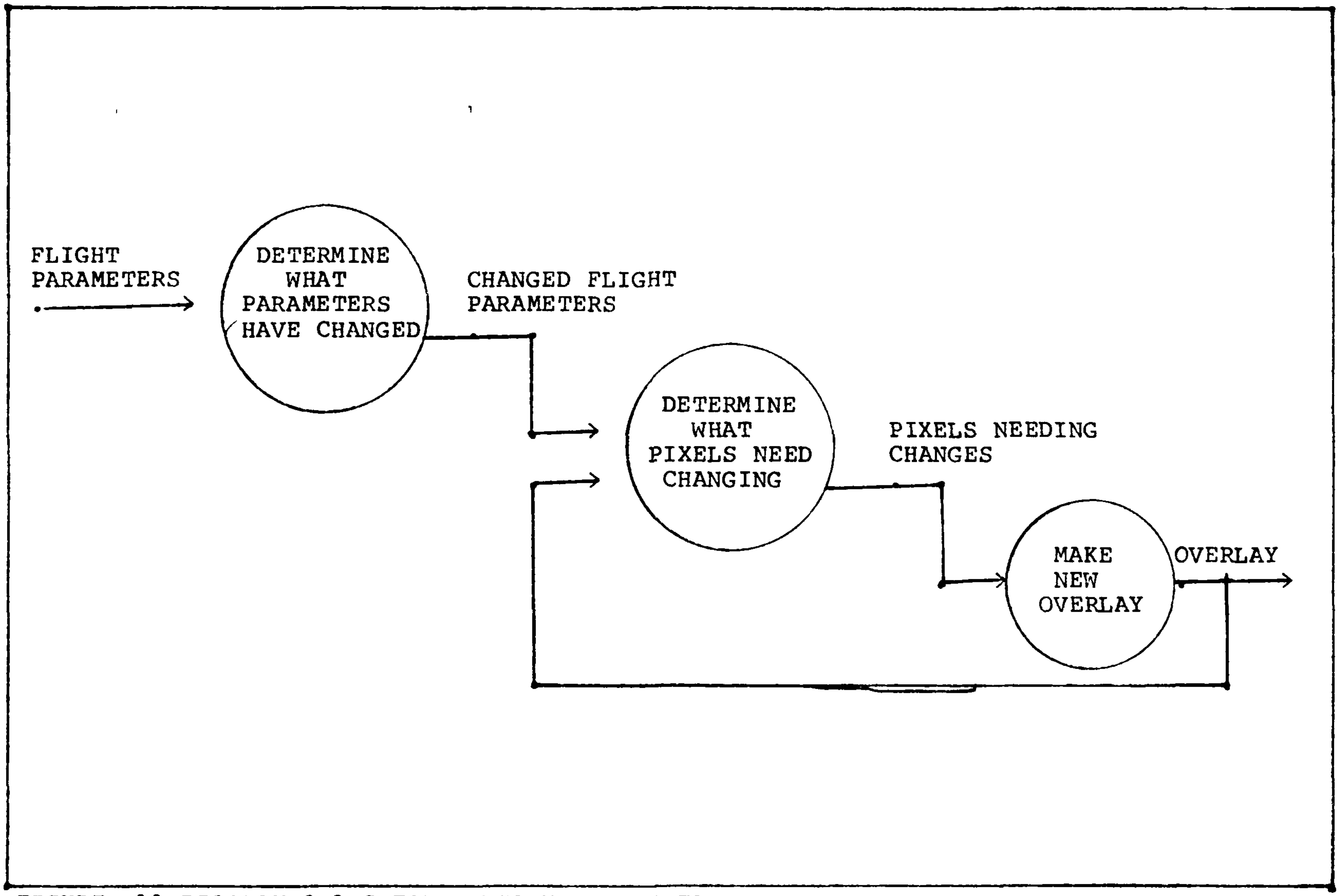

FLIGHT PARAMETERS
DETERMINE WHAT PARAMETERS HAVE CHANGED
CHANGED FLIGHT PARAMETERS
DETERMINE WHAT PIXELS NEED CHANGING
PIXELS NEEDING CHANGES
MAKE NEW OVERLAY
OVERLAY

FIGURE 11 DIAGRAM 2.3 CREATE INFORMATION OVERLAY

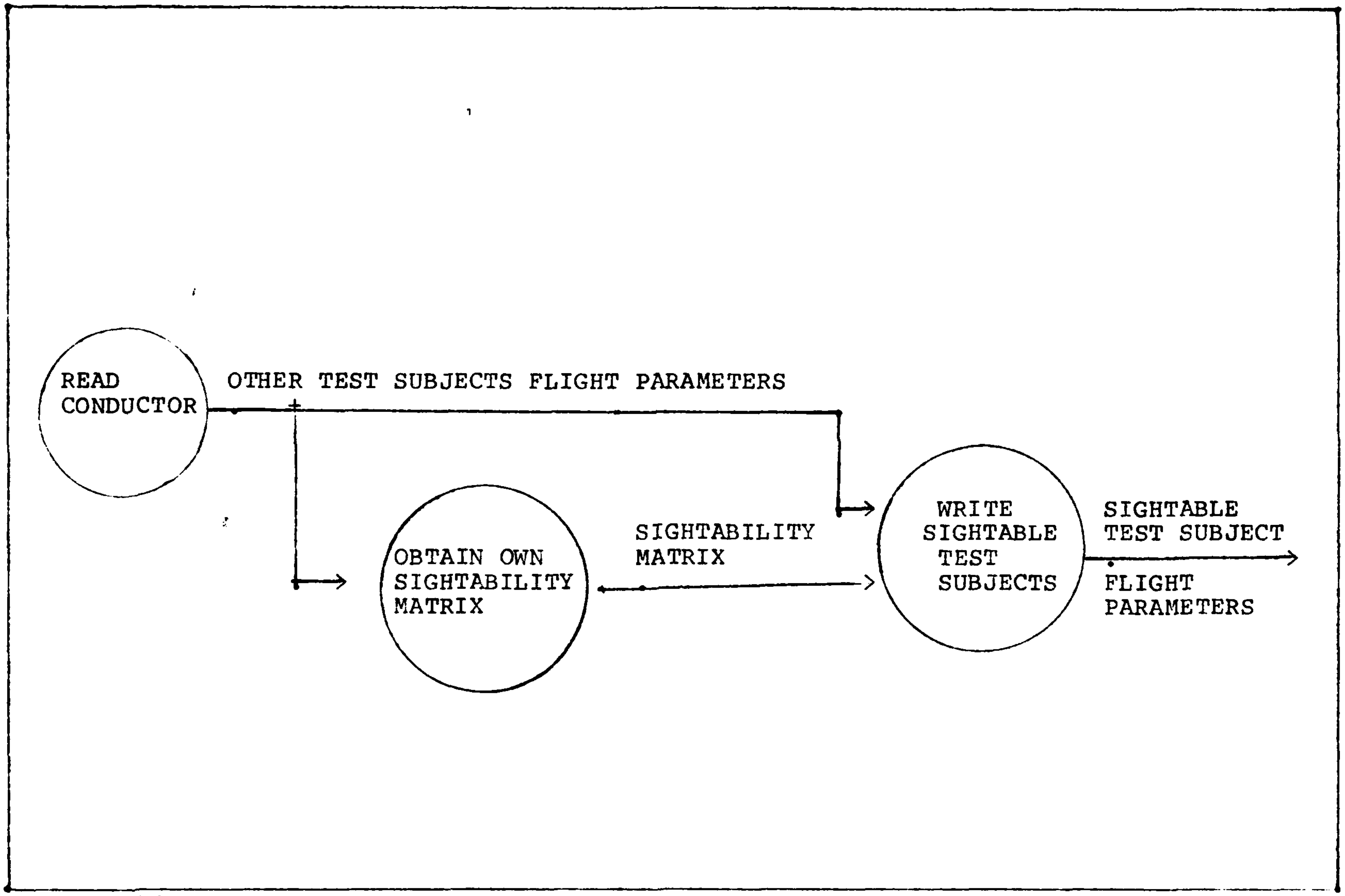

READ CONDUCTOR
OTHER TEST SUBJECTS FLIGHT PARAMETERS
OBTAIN OWN SIGHTABILITY MATRIX
SIGHTABILITY MATRIX
WRITE SIGHTABLE TEST SUBJECTS
SIGHTABLE TEST SUBJECT
FLIGHT PARAMETERS

FIGURE 12 DIAGRAM 2.4 DETERMINE SIGHTABILITY

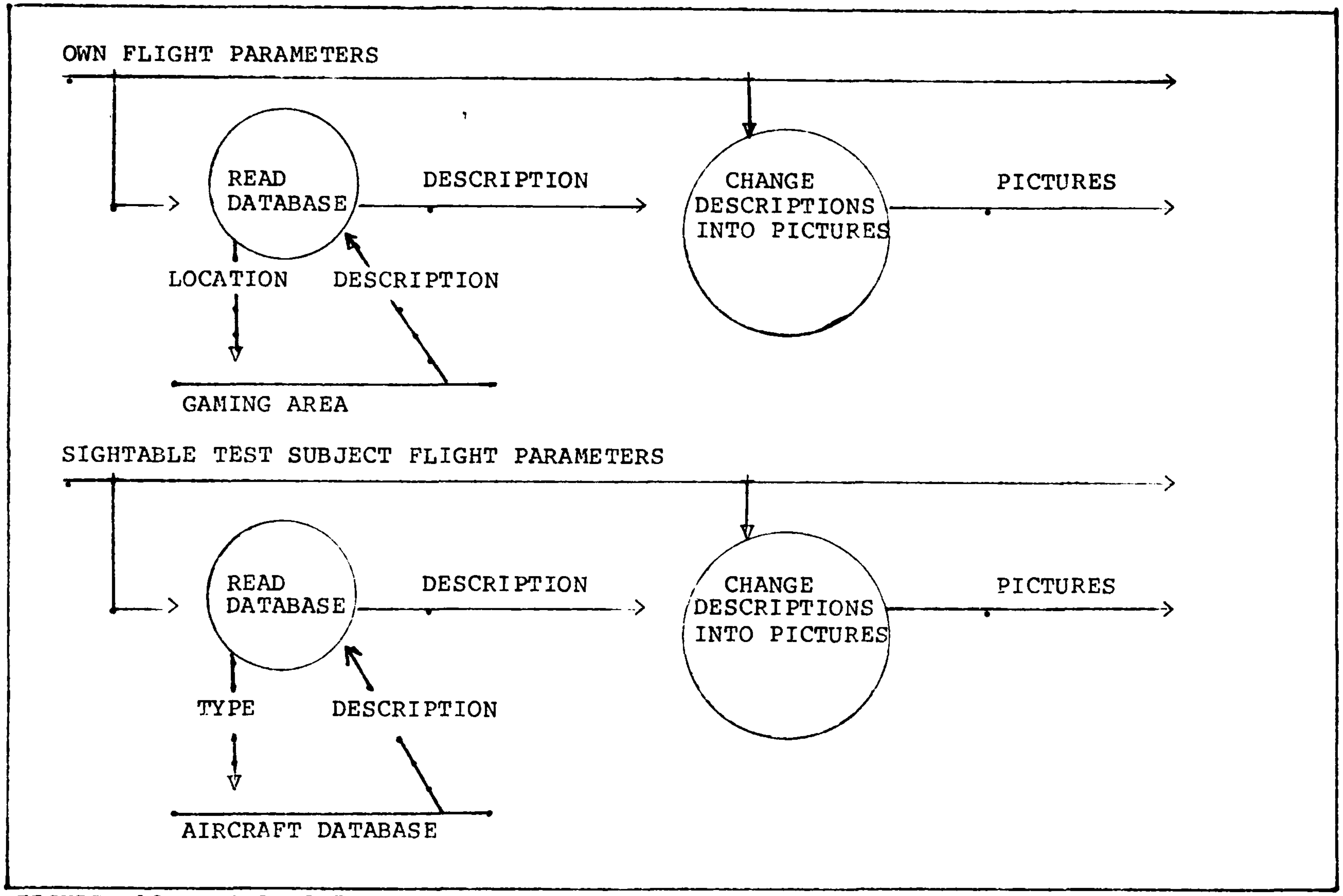


FIGURE 13 DIAGRAM 2.5 RETRIEVE DATABASE

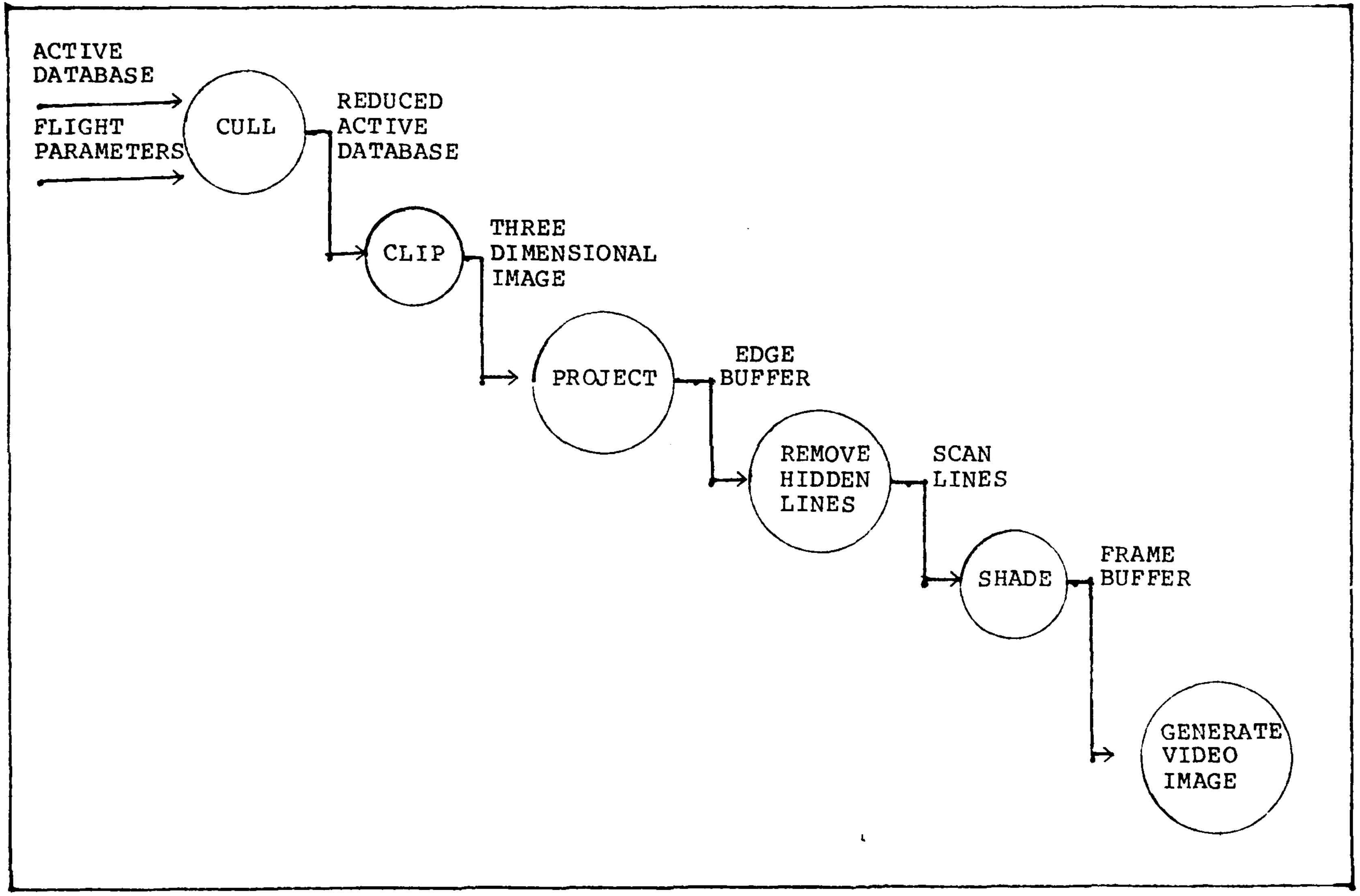


FIGURE 14 DIAGRAM 2.6 GENERATE GRAPHICS DISPLAY

TEST PLAN

The code will be written with a top down modular design with stubs for uncoded packages or procedures. This modularity will enable testing of each part of code as it is completed. The testing will be done to determine that modules are correct and perform within their time constraints.

After each segment of the code is operational, it will be modified to do timing and stress analysis on the code. The timing analysis will consist of writing the time of the processor when it enters and leaves the package. This time difference will then be measured against the amount of time left to perform the image generating process. If the time is not satisfactory then the procedure will be rewritten to improve its performance. If the time cannot be improved, then determine if another procedure or package can be improved.

The stress test will consist of determining what happens when too much data or too little data is introduced into the system. This can happen when the test subject flies outside the gaming area or has nine other aircraft in view while trying to land at the airfield. Testing the effect of setting switches which "do nothing" and determining their effect on the system will be done. Also the analysis will determine what will happen when two test subjects enter the same code into the system. This stress test is needed to ensure that the system is user proof.

SUMMARY

This chapter presented the high level design and the test plan for the PACRAT system. The design covered the conductor's station and the subject's station. The test plan covered how the code will be tested and the associated criteria.

CHAPTER 4

IMPLEMENTATION

INTRODUCTION

The purpose of this chapter is to present how the PACRAT system is implemented. This chapter will discuss the low level implementation of the system, how ADA was used to do some of the detailed work, what had to be done to make the current VAX 11/780 version of ADA work, and how the system should be put together.

The low level implementation will consist of design charts showing how the modules interact with each other and how the packages go together.

The version of ADA, version 1.00, (Ref 14) which was used did not support any floating point operations or any trigonometric functions. These operations and functions had to be implemented. ADA provides "overloading" and this was used to do many of the operations and some functions. Overloading is declaring one function more than once with different interpretations depending on its use. One example of overloading is the "+" operator function. The "+" can be defined to have integer inputs or floating point inputs to produce a floating point output.

The current version of the operating system, iMAX version 1, (Ref 13) does not allow multi-tasking of programs. This is a severe limitation on a system that's main advantage is that if the job needs to be multi-tasked, then adding in more processors can increase the system throughput. With this limitation, the only thing possible to do is to show that each module can do its work correctly in the time allowed. The interface between these modules is slower than expected.

LOW LEVEL DESIGN

CONDUCTOR'S STATION

The operating system interface with the conductor's programs are through a high level program that sets up the systems initializations and executions. This interface is provided by Intel and must be strictly followed for the system to work correctly. From Figure 6 in chapter 3 it is appeārant that there are four functions:

1. Read from the test subjects,
2. Determine the sightability matrix
3. Assemble the conductor output
4. Write to the test subjects

Table IV show the correlation between the attachment number and the bubble found on a data flow diagram or a figure number. Attachment 1 is the interface between iMAX and the program. Its function is to call the next lower routine and is 432 dependent. It is not necessary for the design of the system. In Figure 15 is the function chart of the procedures to be incorporated into the test conductor's initial station. Following that chart is Figure 16 with the functions listed according to packages.

ATTACHMENT NUMBER	FIGURE NUMBER	BUBBLE/FIGURE NAME	ATTACHMENT NAME
1			CONDUCTOR
2	6	TEST CONDUCTOR	CONSITE
3	6.1	READ TEST SUBJECT	CONDUCTOR_WORK
4	7	DETERMINE SIGHT	I_SEE_YOU
3	6.3	ASSEMBLE OUTPUT	CONDUCTOR_WORK
3	6.4	WRITE TEST SUBJECT	CONDUCTOR_WORK

Table IV Correlation between attachemnt and figure numbers

```
-------------------------------------------------------------
|                                                           |
|       CONDUCTOR SIGHT CALCULATIONS                        |
|                                                           |
-------------------------------------------------------------
    |            SUBJECTS |                  | CONDUCTOR |
    |^            WITHIN ||^SIGHTABILITY ||^ DATA    ||
    ||SUBJECT      TEN   V|| MATRIX        V|| OUT     V|
    V  DATA        MILES  V                  |          V
-------------  ---------------- SUBJECT||  -----------
| READ       | |               |  DATA  V| |WRITE      |
|  SUBJECTS  | | I SEE YOU     |         | | CONDUCTOR|
|            | |               |         | |           |
-------------  ----------------          V -----------
                                 ---------
                                 |IN      |
                                 |EQUALS  |
                                 |OUT     |
                                 ---------
```

FIGURE 15 Conductor's Functions Chart

```
-----------------------------------------------------------
|                                                         |
|       CONDUCTOR SIGHT CALCULATIONS                      |
|                                                         |
-----------------------------------------------------------
|PROCEDURE      |PACKAGE        |PACKAGE        |PACKAGE
|               |               |               |
|               +<--------------|---+---------->+
|               |               |   |           |
V               V               V   ^           V
---------------  -------------  ----------  ------------------
| CONDUCTOR   |  | CONDUCTOR |  | I      |  |                |
|  SIGHT CALC |  | WORK      |  | SEE    |  | REAL OPERATIONS|
|             |  |           |  | YOU    |  |                |
---------------  -------------  ----------  ------------------
|PROCEDURE      |CONTAINS       |PROCEDURE      |PROCEDURES
|               |PROCEDURES     |               |
|               |AND            |               |
V               |DATA           V               V
----------------             -------------  ----------------
| READ_SUBJECT |             |           |  |              |
| SQUARE_ROOT  |             | I_SEE_U   |  | ASIN         |
| I_SEE_U      |             |           |  | SQUARE_ROOT  |
| WRITE_       |             |           |  |              |
|   CONDUCTOR  |             |           |  |              |
----------------             -------------  ----------------
                                |PROCEDURE
                                |
                                V
                             -------------
                             |           |
                             | ASIN      |
                             |           |
                             -------------
```

FIGURE 16 Conductor's Packages Chart

TEST SUBJECT'S STATION

The operating system interface with the subject's programs are through a high level program that sets up the system's initializations and executions. This interface is provided by Intel and must be strictly followed for the system to work correctly. From Figure 8 in chapter 3 it is appearant that there are six functions:

1. Produce starting parameters
2. Generate location parameters
3. Create Information overlay
4. Determine the sightability
5. Retrieve the data base
6. Generate the graphic display

In implementing these functions it became necessary to move some of these functions into a lower level to improve the amount of parallelism in the programs. Creating of the information overlay can be done while creating of the location parameters since its only input is the location parameters. The retrieval of the data base is only done when generating the graphic display, so the subject's high level is simplified and the parallelism is more prevalent. Table V shows the correlation between the figure numbers of chapter 3's design and the implementation's attachments. The top level is for iMAX use and does not effect the design of the system. Figure 17 shows how figure 8 was implemented and figure 18 shows all the packages used in the test subject's station and how they fit together.

ATTACHMENT NUMBER	FIGURE NUMBER	BUBBLE/FIGURE NAME	ATTACHMENT NAME
5			TOP OF SUBJECT
6	8	TEST SUBJECT	SUBJECT
7	9	STARTING PARAMETER	INITIAL TEXT
8	10	GENERATE LOCATION	EQUATION FLIGHT
	11	INFORMATION OVERLAY	
	13	RETRIEVE DATA BASE	
9	12	DETERMINE SIGHTABLE	SIGHT DESCRIPTION
	13	RETRIEVE DATA BASE	
10	14	GENERATE DISPLAY	PICTURE
11	10	GENERATE LOCATION	EQOATION MOTION
12	11	INFORMATION OVERLAY	INFORMATION OVERLAY
13	13	RETRIEVE DATA BASE	ACTIVE DATA BASE

Table V Correlation between figure and attachment numbers

```
--------------------------------------------------------------
|                                                            |
|               SUBJECT                                      |
|                                                            |
--------------------------------------------------------------
|^INITIAL                LOOPING
||PARAMETERS                |
||ID AND        ------------+-----------------
| TYPE          |CONCURRENT|                |
|               |          |                |
|               |          |                |
V               V          V                V
---------  -------------  -----------  ----------
|INITIAL |  |DETERMINE   |  |EQUATION |  |MAKE     |
|TEXT    |  |SIGHTABILITY|  |of FLIGHT|  |PICTURES |
---------  -------------  -----------  ----------
```

FIGURE 17 Subject's Function Chart

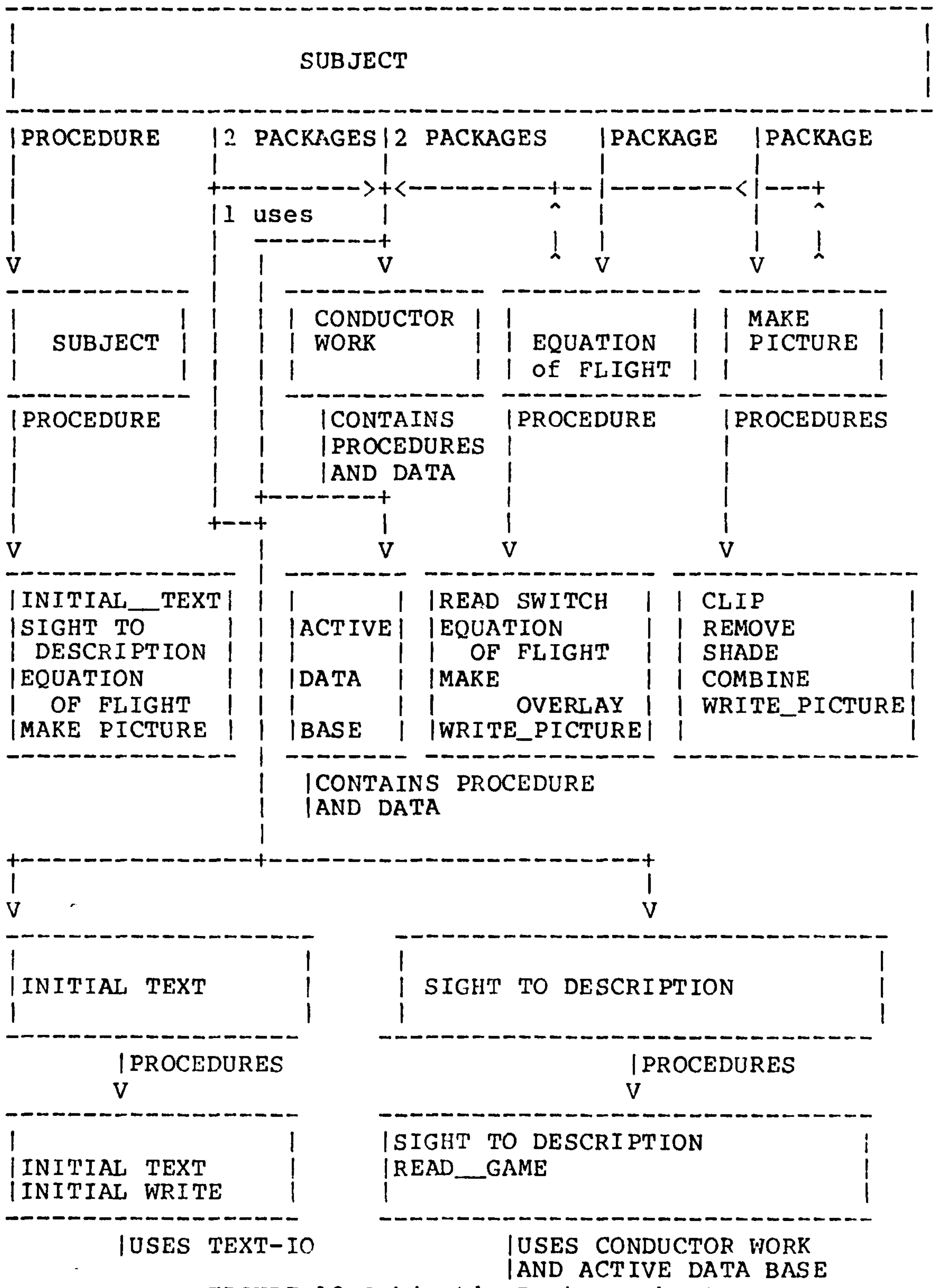


FIGURE 18 Subject's Package chart

MISCELLANEOUS FUNCTIONS

The EQUATION OF MOTION procedure needs floating point operations. Since the initial version of ADA did not support floating point operations, it was necessary to create these operations. Various operations were added as they were needed. This package, REAL_OPS, is found in attachment 14. The specification portion of the package lists all the functions needed; more can be added when they are required.

Most routines that did any calculations needed to use a trigonometric function. Since the initial version of ADA did not include the trigonometric functions it was necessary to create these functions. The algorithms for these functions were found in Schaum's outline series (Ref 1). Due to speed requirements these functions do not check for proper input and thus may give wrong results. The package is found in attachment 15 with the specifications giving which functions are implemented.

To implement integer input and output, the package INTIO was produced. Integer IO uses the Intel provided Text IO to accomplish the function.

Table VI shows what attachment numbers are used for miscellaneous functions. All of these functions should disappear when a completed version of ADA appears.

attachment number	attachment name
14	real operations
15	trigonometric functions
16	integer io

Table VI Attachment number for micellaneous functions

SUMMARY

The low level design and implementation has been presented. The test conductor was implemented with four high level programs. The test subject was implemented with seven high level programs. The ADA code can be found in the attachments.

CHAPTER 5

RESULTS

INTRODUCTION

The results of this thesis effort did not completely meet the objectives as described in chapter 2. Test plan results should have been included as outlined in Chapter 3. The reason that the results are not being reported is due to the Intel iAPX 432 system configuration. The Intel iAPX 432 requires an Intel Series III to talk to the outside world in its present configuration. The Institute received the 432 in mid June and at the time of writing does not have a Series III. Because of this unfortunate event, the results reported here will be from writing the code and compiling the ADA code.

The results will be broken into three sections:

1. The compiler and its features.
2. The operating system supplied and its features.
3. The results of compiling the programs written.

The COMPILER

The compiler used was the version 1.00 compiler for the iAPX 432 Cross Development System. There is a lot of work left to be done to make that compiler a full scale ADA compiler. It was interesting that some of the parts left out of the compiler were easily implemented in assembly code. Parts of the floating point operations were implemented.

The input/output using the package Text_io left a lot to be desired. There is no Direct Memory Access, DMA, for reading and writing between the Data Processors and the Attached Processors. The programs in Conductor_Work which read and write the data should be rewritten to improve their speed as soon as there exists the capability as outlined in the iMAX version 2 manual (Ref 13). The present compiler does not allow the writing of DMA I/O.

When the compiler comes out with the floating point operations implemented, it would be advisable to go through all the packages that use the Real_ops package and change those packages to use the compiler floating point operations. The reason is twofold. One, the floating point was implemented using a second address in memory. This is slower than using the stack. Second, the implementation was not using the pragma INLINE which should have been used. This is because the compiler would not allow the use of this pragma.

The mathematical package with the trigonometric functions should be checked out to see if it responds fast enough. If the functions are not fast enough then they should be rewritten to use a table look-up scheme or to reduce their accuracy. If they are fast enough, then it might be possible to add in-bounds checking into the routines. When Intel provides their package, times for execution should be checked out to determine which is fastest and to use the fastest routines.

The OPERATING SYSTEM

The operating system which is now available at the Institute is the initial version of iMAX. This version does not allow the use of tasking of sub-processes. When the operating system will allow tasking of modules then the full capability of the system will be seen. Until then a backward system is to start a process and let it wait till it is sent a message. When it has a message the process will work on that message until it is done and then go back to wait. The drawback to this is that the operating system will not allow a procedure to "go to sleep" and then wake up. The operating system will improve and when it does segments of code that need to have the tasking added can do so.

The operating system also is a reason for the poor primatives of the input/output. This should improve with version 2, but this remains to be seen.

The PROGRAMS

The programs written so far have been small and as mentioned before need to be worked on when new releases of the compiler and the operating system arrive. The structure of the programs has been laid out in Chapter 4 and should be easily verified for the sponsor when the system is ready to be used.

The programs were written to just load them on to the Intel iAPX 432 and test out the system. Hopefully there will not be too many errors when it is finally loaded on to the system and run.

The DATABASE

The visual data base that the Singer-Link Corporation sold to the Air Force as part of the F-111 DIG is to be used as the 600 square mile gaming data base in PACRAT. This data base was acquired in internal machine format on a Singer-Link operating system file format. Another part of this thesis effort was to read this tape and make sense out of the numbers. Using the AFIT Scientific Support Computer, a VAX 11/780 running the UNIX operating system, the numbers were changed from a machine readable form to decimal format using the UNIX functions device dump and octal dump and a program to change octal numbers to decimal numbers. The data was backed off the disks by use of the "TAR" program onto tape 2026 which resides in the AFIT tape library. Being able to read the data base does not mean there is a usable data base. The numbers now must be interpreted correctly. The description of the numbers are found in a video tape provided by the Singer-Link Corporation. Since the video tape has not arrived yet the author has not viewed the tape and thus cannot interpret the numbers correctly.

SUMMARY

The results have been from compiling and coding the programs necessary for this design of the PACRAT system to function on the Intel 432. The actual execution on the 432 was not done due to lack of equipment. The compiling and linking of programs was done on a VAX 11/780 under VAX/VMS operating system. The compiling and linking were completely documented and easy to execute but took an excessive amount of cpu time to complete the work. This could be because of the size of the system or the number of programs compiled. The design work was straight forward and going from the requirements to the design and finally to the code was a simple process.

CHAPTER 6

SUMMARY and RECOMMENDATIONS

SUMMARY

Part of this thesis effort was to review the requirements (Ref 23, 26) and to implement the system from those requirements. The requirements were summarized in Chapter 2 in a table and in SADT charts. There were no major changes from the initial work (Ref 23). Thus the changes were in numerical values, not in concept. The high level design was explained in chapter 3 using Data Flow Diagrams. Chapter 4 showed how the design was structured. The results of the implementation were presented in Chapter 5.

The high level design presented in Chapter 3 can be directly correlated to the requirements in Chapter 2 by comparing the SADT charts and the Data Flow Diagrams.

The implementation was done in ADA based upon the DFDs. Chapter 4 describes the structure of the implementation. The equations of motion were obtained from AFWAL/FIGD in FORTRAN and the author did a straight conversion to ADA without any logical restructuring, although ADA has some features (Ref 31,36) which could enhance the current program. These features were not implemented in the version of the compiler that was used. All the other programs were the author design and are described in Chapter 4.

RESULTS

The results of this thesis effort did not completely meet the objective as described in Chapter 2. Test plan results should have been included as outlined in Chapter 3. The reason for the omission is that the hardware did not arrive in time. The Intel 432 micromainframe computer requires an Intel Series III to act as its interface to the outside world. This is because the interface processor does communicate with the outside world, but just follows the user's commands. Without the Intel Series III, the Intel 432 could not do any work in this effort. The results in Chapter 5 were from the design and implementation, not from execution of programs as the author had wished.

The visual data base that the Singer-Link Corporation sold to the Air Force as part of the F-111 DIG is to be used as the 600 square mile gaming data base in PACRAT (Ref 26). This data base was acquired in internal machine format on a Singer-Link operating system file format. Another part of this thesis effort was to read this tape and make sense out of the numbers. Using the AFIT Scientific Support Computer, a VAX 11/780 running the UNIX operating system, the numbers were changed from a machine readable form to decimal format.

The data was transferred from the disks by use of the "TAR" program onto tape 2026 which resides in the AFIT tape library. Being able to read the data base does not mean there is a usable data base. The numbers now must be interpreted correctly. The description of the numbers are found in a video tape provided by the Singer-Link Corporation. Since the video tape has not arrived yet the author has not viewed the tape and thus cannot interpret the numbers correctly.

RECOMMENDATIONS

Based on the work presented in this thesis, the amount of work still to be done on the Intel 432, and the amount of work still to be done on PACRAT, the following recommendations are presented for areas of future study:

1. Data base thesis. The visual data base is a 600 square mile area and it has 7.6 million bytes of data stored in it. The thesis would take the current data base and transform it into a data base such that the PACRAT system could have rapid access to it. This would encompass determining what type of data base, creating the data base, and providing for access from the Intel 432.

2. Follow - on project effort. The follow - on project effort would entail testing the code on the Intel 432 and correcting any mistakes found.

Hopefully this would be a minimal effort and would just familarize the student with ADA and the Intel 432. The student could code an input/output routine to test the trigonometric functions and thus familarize the student with the current disadvantage of the Intel 432, its I/O.

3. A network project. Since the PACRAT is a large number of small computers linked together, a project for the advanced network class would be to take the requirements set forth and design the protocols necessary to make the network processors know the current situation.

The following are not areas of study but helpful things to watch out for when using the Intel 432.

1. The ADA compiler version 1.00 should be replaced with version 1.01 or later release. The version 1.00 compiler does a copy of the arguments and does not send the access descriptor. The floating point operations should be implemented but are not in version 1.00. DMA transfers should be able to be coded as described in iMAX version 2.01 documentation. These should be corrected in later releases.

2. The operating system iMAX version 1.00 should

be replaced with version 2.01 or later release.

3. Review Appendix A on the Intel iAPX 432 architecture.

DISCUSSION

The main objective of this thesis effort remains unobtained, how well the Intel 432 works and does it meet the sponsor's goals. The main area of concentration for further study of the Intel iAPX 432 should be in the area of the operating system. If the areas of further study are carried out, the sponsor will have this information and the Institute will have a powerful small computer.

From the writing and compiling of the ADA code the author would advise the sponsor to continue acquiring the Intel 432 and its associated software as new releases become available and delay committment to the 432 until:

(1) The compiler is a fully qualified version of ADA.

(2) The 432 system has some form of easy mass data transfer into and out of the data processors.

(3) The 432 chip set is finalized and mass produced. This is because the extreme amount of change the chip set is experiencing and one compiler is needed for each different chip version.

To improve the Intel 432's performance with PACRAT it is advisable to:

(1) After the memory and data bus chips are out and available for study, have a thesis effort to decide if this added capability will be necessary for the requirements in PACRAT.

(2) Interleave the memory to increase the performance. Intel documents that the interleaving of memory can increase performance by 30 to 40 percent (Ref 17,41).

The author believes that a 432 with four general data processors can perform the required computations of the reduced simulator as described in this thesis.

BIBLIOGRAPHY

1. Ayres, Frank Jr., Theory and Problems of Differential and Integral Calculus, Mcgraw-Hill Book Company, 1964, pg 242

2. Bayliss et. al., The Instruction Decoding Unit for the VLSI 432 General Data Processor, IEEE Journal of Solid-State Circuits, October 1981, Pg 531

3. Bayliss et. al., The Interface Processor for the VLSI 432 32-Bit Computer, IEEE Journal of Solid-State Circuits, October 1981, Pg 522

4. Beyers et. al., A 32-Bit VLSI CPU Chip, IEEE Journal of Solid-State Circuits, October 1981, Pg 537

5. Budde et. al., The Execution Unit for the VLSI 432 General Data Processor,IEEE Journal of Solid-State Circuits, October 1981, Pg 514

6. Budde, David et.al., 32-bit Computer Execution Unit, 432 Architecture Workshop version 2.5, Sept 1981, pg A-34

7. Cox et. al., A Unified Model and Implementation for Interprocess Communication in a Multiprocessor Environment, Proceedings of the Eight Symposium on Operating System Principles, Dec 1981, Pacific Grove, California, ACM, pg 125

8. Cox, George et.al., Interface Processor for the 32 Bit Micromainframe Computer, 432 Architecture Workshop version 2.5, Sept 1981, pg A-45

9. Cox, George et.al., Supporting Ada Memory Management in the iAPX 432, Symposium on Architectural Support for Proramming Languages and Operating Systems, March 1-3, 1982, Palo Alto California, ACM, pg 117

10. Hemenway, Jack, Object-Oriented Design Manages Software Complexity, EDN, August 19,1981, pg 141

11. Hemenway, Jack and Grapple, Robert, Understand the Newest Processorto Advoid Future Shock, EDN, April 29, 1981, pg 129

12. iAPX 432 Interface Processor Architecture Reference Manual, Intel 1981

13. iMax 432 Reference Manual, Intel 1981

14. Intel 432 CDS VAX/VMS Host User's Guide, Intel 1981

15. Intel 432 CDS Workstation User's Guide, Intel 1981

16. Introduction to the iAPX 432 Archecture, Intel Corp. 1981

17. Introduction to the Intel 432 Cross Development System, Intel 1981

18. Janway, Brad, Intel Corp. Personnel phone conversation, 14 June 1982.

19. Kahn, Kevin C. and Pollack, Fred, An Extensible Operating System for the Intel 432, COMPCON 81, San Francisco, Calf. February 23-26 1981, pg 398

20. Kahn et. al., iMAX: A Multiprocessor Operating System for an ObjectBased Computer, Proceedings of the Eight Symposium on Operating System Principles, Dec 1981, Pacific Grove, California, ACM, pg 127

21. Kaminker et. al., A 32-Bit Microprocessor with Virtual Memory Support, IEEE Journal of Solid-State Circuits, October 1981, Pg 548

22. Kartashev, Svetlana and Kartschev, Steven, Adaptable System with Dynamic Architecture, National Compute Conference 1981, Pg 111

23. Knight, Donald P. Design of a Microprocessor-Controlled EXperiment TestStation with Real Time Computer Generated Out-The-Window Cockpit Images,Thesis, Wright Patterson Air Force Base, Ohio, December 1981.

24. Lehmann, Thomas, Notes from 432 Architecture Workshop, Intel Corp.Chicago, Ill., Jan 1982.

25. Mazor and Wharton, Compact Code - iAPX 432 Addressing Techniques, ComputerDesign, May 1982, Pg 249

26. McKinley, Richard, sponsor,AFAMRL/BBA, Biomedical Engineer, (personal discussion December 1981).

27. Mikklson et. al., An NMOS VLSI Process for Fabrication of a 32-Bit CPU Chip, IEEE Journal of Solid-State Circuits, October 1981, Pg 542

28. Passafrume, John, ADA Education for Technical Managers, 3 March 1981, DTIC ADA97524

29. Patterson, John, Extending ADA into Silicon, Defense Electronics, September 1981

30. Pollack et. al., The iMAX-432 Object Filing System, Proceedings of the Eight Symposium on Operating System Principles, Dec 1981, Pacific Grove, California, ACM, pg 137

31. Pyle, Ian C., The ADA Programming Language, Prentice-Hall International 1981

32. Ratterner, Justin et.al., A Methodolgy for VLSI Chip Design, Lambda,Second Quarter, 1981, pg 34

33. Rattner, Justin et.al., A 32-Bit VLSI MicroMainframe Computer Survey, 432 Architecture Workshop version 2.5, Sept 1981, pg A-14

34. Rattner, Justin and Lattin, William, Ada Determines Architecture of 32-bit Microprocessor,Electronics,February 24,1981,pg 119

35. Ratterner,Justin and Cox, George, Object-Based Computer Architecture,Computer Architecture News, October 15, 1980, pg 4

36. Reference Manual for the ADA Programming Language, DOD, July 1980

37. Reference manual for the Intel 432 extensions to ADA, INTEL 1981

38. Richardson, William et.al., The 32b Computer Instruction Decoding Unit, 432 Architecture Workshop version 2.5, Sept 1981, pg A-24

39. Rutledge, James Lt Col, Notes from Software Engineering course, AFIT,Wright Patterson Air Force Base, Ohio, Air Force Institute of Techmology,Summer 1981.

40. Schindler, Max, Ada Teams Up With 32-bit chip to form efficiecnt OEM system. Electronic Design,February 19,1981, pg 36

41. System 432/600 System Reference Manual, Intel 1981

42. Tyner, Paul, iAPX 432 GDP Architecture Reference Manual, Intel Corp.Aloha, Oregan, Jan 1981.

43. Zeigler et. al., ADA for the Intel 432 Microcomputer, Computer, June 1981,Pg 47

44. Zeigler, Stephen et al, The Intel 432 Ada Programming Environment , COMPCON 81, San Francisco, Calf. February 23-26 1981,pg405

APPENDIX A

THE INTEL iAPX 432 MICROMAINFRAME

TABLE OF CONTENTS

INTRODUCTION

Intel first released the concept of a computer using an object based architecture in October, 1980 (Ref 35). Object based architecture is a style of architecture which has the ability to raise the level of hardware/software interface and integrate the concept of data abstraction, domain based protection and high level system functionality (Ref 36). Some new ideas from this architecture include capability based addressing (Ref 11), lexical level addressing (Ref 9), activation records (Ref 24), self defining data types (according to what type of access description is pointing to it) (Ref 24), language directed architecture (Ref 34), storage to storage operations (Ref 24,42), stack used for expression evaluation only (Ref 42), and it lowers the semantic gap (Ref 29,34). This appendix will tell how an object based computer puts these new ideas into pratice.

This appendix will discuss the Intel iAPX 432 Micromainframe and how it is an object based computer. The objects will be defined and all various types examined. Instructions and memory management will be discussed followed by description of the system interactions.

OBJECTS

There are two parts to an object: data and pointers. This difference in parts of objects causes the machine to notice the difference between data and pointers(Ref 9). The users are allowed to create their own types which will be treated differently from all other types. The operating system creators did this in writing iMAX, the operating system, so system tables are different from other types of objects(Ref 20). All objects are at most 64K bits long but most will be only about 200 bytes long (Ref 30). Below the various types of objects will be discussed. The size and number of objects could affect system performance.

PROCESSOR OBJECT

There is one processor object for each general data processor in the system. The object contains the following pertinent information about the processor(Ref 24,42): faults, the status of the processor, ie sleeping, working, waiting or faulted and various details about the system, eg top of memory, number of processors, etc. There is a queue pointer that points to a list of waiting processes for the processor. The processor object has a pointer to the current process object that it is executing. See Figure 1 for a view of the processor access list and Figure 2 for the processor data.

```
 ----------------------------------------
| AD to Carrier to Diagnostic Port      | 19
 ----------------------------------------
| AD to Carrier to Reconfiguration Port | 18
 ----------------------------------------
| AD to Carrier to Alarm Port           | 17
 ----------------------------------------
| AD to Carrier to Normal Port          | 16
 ----------------------------------------
| AD to Diagnostic Port                 | 15
 ----------------------------------------
| AD to Reconfiguration Port            | 14
 ----------------------------------------
| AD to Alarm Port                      | 13
 ----------------------------------------
| AD to Normal Port                     | 12
 ----------------------------------------
| AD to Surrogate Carrier               | 11
 ----------------------------------------
| AD to Current Carrier                 | 10
 ----------------------------------------
| AD to Current Port                    |  9
 ----------------------------------------
| AD to Current Message                 |  8
 ----------------------------------------
| AD to Delay Carrier                   |  7
 ----------------------------------------
| AD to Delay Port                      |  6
 ----------------------------------------
| AD to Processor Carrier Object        |  5
 ----------------------------------------
| AD to Object Table Dictionary         |  4
 ----------------------------------------
| AD to Global Communication Segment    |  3
 ----------------------------------------
| AD to Local Communication Segment     |  2
 ----------------------------------------
| AD to Current Process Carrier         |  1
 ----------------------------------------
| AD to Processor Data Segment          |  0
 ----------------------------------------
0                                      32
```

Figure 1 Processor Access Descriptor (AD) List

```
-----------------------------------------
|   Processor and Process               | 71
|  Fault Information Area               |
|                                       | 32
-----------------------------------------
|                                       | 31
|       REVERSED                        |
|                                       |  2
-----------------------------------------
| Processor Status                      |  1
-----------------------------------------
| Object Lock                           |  0
-----------------------------------------
0                                       16
```

Figure 2 Processor Data Segment

<u>PROCESS OBJECT</u>

There is one process object for each process. There can be many process objects for one processor object. Process objects point to the same class of information that the processor object point to but a different level of data is stored(Ref 24,42). The process object also points to the global variables of the process. Instead of pointing to a queue of process objects the process objects points to a queue of context objects. The process object also points to the next process object in the queue. See Figure 3 for a view of the process access list and Figure 4 for the process data segment.

```
 ----------------------------------------
| AD to Surrogate Carrier                | 11
 ----------------------------------------
| AD to Current Carrier                  | 10
 ----------------------------------------
| AD to Current Port                     |  9
 ----------------------------------------
| AD to Current Message                  |  8
 ----------------------------------------
| AD to Fault Port                       |  7
 ----------------------------------------
| AD to Scheduling Port                  |  6
 ----------------------------------------
| AD to Dispatching Port                 |  5
 ----------------------------------------
| AD to Process Carrier                  |  4
 ----------------------------------------
| AD to Local Object Table               |  3
 ----------------------------------------
| AD to Global Access Segment            |  2
 ----------------------------------------
| AD to Current Context                  |  1
 ----------------------------------------
| AD to Process Data Segment             |  0
 ----------------------------------------
0                                       32
```

Figure 3 Process Access Descriptor (AD) List

```
 ----------------------------------------
|    Stroage Block Descriptor            | 15
|                                        |
|                                        |  8
 ----------------------------------------
|       REVERSED                         |  7
|                                        |  6
 ----------------------------------------
| New AD Copy                            |  5
|       used in faulting                 |  4
 ----------------------------------------
| SRO Lock                               |  3
 ----------------------------------------
| Current Block Index                    |  2
 ----------------------------------------
| Beginning Block Index                  |  1
 ----------------------------------------
| Object Lock                            |  0
 ----------------------------------------
0                                       16
```

Figure 4 Process Data Segment

CONTEXT OBJECT

There are many context objects for each Process. The context data(Ref 24,42) holds the expression evaluation stack, trace information, and variable data. This variable data is part of the lexical level addressing scheme. The context access list is equivalent to an activation record. It contains a pointer to the context data, a pointer to the domain object, 4 pointers to context objects, pointers to message objects, a pointer to the operand stack, and a pointer to the constant data segment. See Figure 5 for a view of the Context Access List and Figure 6 for the Context Data Segment.

```
---------------------------------------
|                                     |
|  (WORKING ADs                       |
|                                     |  10
---------------------------------------
| AD to Context's Operand Stack       |  9
---------------------------------------
| AD to Domain of Definition          |  8
---------------------------------------
| AD to Entry Access Segment 3        |  7
---------------------------------------
| AD to Entry Access Segment 2        |  6
---------------------------------------
| AD to Entry Access Segment 1        |  5
---------------------------------------
| AD to Current Context               |--4-->|
---------------------------------------      v
| AD to Message Object                |  3   |
---------------------------------------      |
| AD to Previous Context              |  2   |
---------------------------------------      |
| AD to Constants Data Segment        |  1   |
---------------------------------------      |
| AD to Context Data Segment          |  0   v
---------------------------------------<-----|
0                                    32
```
Figure 5 Context Access Descriptor (AD) List

```
-----------------------------------------
|                                       |
|                                       |
|                                       |  7
-----------------------------------------
| Trace Event Code                      |  6
-----------------------------------------
| Trace IP                              |  5
-----------------------------------------
| Trace Object Index                    |  4
-----------------------------------------
| IP (instruction Pointer) -- in Bits   |  3
-----------------------------------------
| Current Instruction Object Index      |  2
-----------------------------------------
| SP (Stack Pointer)                    |  1
-----------------------------------------
| Context Status                        |  0
-----------------------------------------
0                                      16
```
Figure 6 Context Data Segment

DOMAIN OBJECTS

The domain object is Passed from one context to another thus creating the hidden data or process concept. The domain data is also the constant data segment. The domain data is pointed to by the context access list and the domain access list. The domain access list(Ref 24,42) points to the domain data segment, the instruction segments and other domain objects. See Figure 7 for a view of the domain access list.

```
---------------------------------------------
|                                           |
|   (More User's ADs)                       |
|                                           |
---------------------------------------------
| User's AD                                 |   4
---------------------------------------------
| User's AD                                 |   3
---------------------------------------------
| User's AD                                 |   2
---------------------------------------------
| User's AD                                 |   1
---------------------------------------------
| User's AD                                 |   0
---------------------------------------------
0                                          32
```

Figure 7 Domain Access Descriptor (AD) List

INSTRUCTION SEGMENTS

Intructions only reside inside instruction segments. Instructions are bit addressable and can be from 5 bits long to 315 bits long(Ref 24,42,38). An average line of ADA code translates into 32 bits of assembled code(Ref 24). Instruction segments rights are read and execute only. See Figure 8 for view of the Instruction Segment.

```
 ----------------------------------------
|                                        |
|   Instructions                         |
|                                        |  7
 ----------------------------------------
| Trace Object Index in Domain           |  6
 ----------------------------------------
| Fault Object Index in Domain           |  5
 ----------------------------------------
| Data Constants Segment Index           |  4
 ----------------------------------------
| Displacement of Initial Instruction    |  3
 ----------------------------------------
| Operand Stack Segment Length           |  2
 ----------------------------------------
| Context Data Segment Length            |  1
 ----------------------------------------
| Context Access Segment Length          |  0
 ----------------------------------------
0                                      16
```
Figure 8 Instruction Segment

INSTRUCTIONS

Instructions have four fields(Ref 24,42,38,25): class, format, references, and operation code. All fields are variable in length by use of Huffman codes. The class field tells the system how to interpret the other fields, it is 4 or 6 bits in length. The format field informs the system about the references to follow. The format field is from 0 to 4 bits long. There can be from 0 to 3 references. A reference can be from 11 bits to 100 bits depending on the addressing scheme. Operand codes vary from 0 to 5 bits long and carry the standard information to the system. Figure 9 shows a layout of the instruction format.

```
MSB                                    LSB
------------------------------------------
| Opcode | Reference(s) | Format | Class |
------------------------------------------
```

Class must be present
MSB - Most Significant Bit
LSB - Least Significant Bit

Figure 9 Generalized Instruction Format

The format can be one of five types. The first type is no references; this is typical of a return instruction. The second type is one reference; this is typical of a set a location to zero. The third type is two references with two data operands; this is typical of equate A to B. The fourth type is two references with three data operands; this is typical of equate A to A plus B. The fifth format is with three references; this is a typical instruction; equate A to B plus C. Each reference can be to the top of stack, a scalar variable or an indexed variable. These can be either direct or indirect references.

The Instruction set contains all the standard operations plus those necessary for interprocess communication. No instruction can reference a register because no registers exist at the assembly level. The machine level code is ADA, thus making it a language directed architecture(Ref 29,34).

MEMORY

Memory(Ref 9,24,30,41) is stored as 39 bits, with 32 bits of data and 7 bits of error correcting codes, one error corrected and two errors detected. The data bus is 36 bits wide, one parity bit for each 8 bits of data. The memory can be up to 2 **24 bits long, but it is segmented. There can be 2**24 segments. Each segment can be up to 2**16 bits long. This yields a virtual memory of 2**40 or 1 terabyte of memory. Associated with each segment are 8 bits of rights information. These bits are used in a capability based addressing scheme. As mentioned earlier, the context access list has 4 pointers to context access lists. These pointers are actually a table on chip, the compiler must keep track of the access list pointers. These pointers yield the lexical level addressing schemes. Segment numbers are 16 bits long. The address of the segment is 24 bits, these 24 bits are a physical location not a virtual location. Memory can be referenced as a scalar in 11 bits. The segment can be off chip and must be loaded from a table in memory, this takes 17 bits of address. Finally the entire context object might have to be reloaded and this takes 32 bits of data.

There are many data types available(Ref 24,42,6). The shortest data type is the bits which can vary from 1 to 32 bits long. The next longest type is the characters, 7 bits of ASCII code. The other types of data are all numerical in nature. Ordinals are just unsigned integers. Short integers are 16 bits long. Long integers are 32 bits long. Short reals are 32 bits of data. Long reals are 64 bits of data. Temporary reals are a full 80 bits long. All real computations are done in temporary real format. Remember instructions are not data types and instructions range from 5 to 315 bits long.

Memory can be interleaved to yield an increase of efficiency from 30 to 40 percent(Ref 17,41).

PERFORMANCE

The Intel 432 indicates (Ref 24,41) a very high instruction rate. Intel admits that the instruction rate is determined by the number of busses. If there are a sufficient number of busses, the speed of the 432 can be increased by simply inserting another 432 general data processing board. One bus can handle up to 4 boards before any more boards will not improve performance. 2 busses can handle up to 9 boards. Without memory interleaving, 4 general data processors are equivalent to an IBM 370/148 at

2 million instructions per second(Ref 24), 9 boards are equivalent to 4 or 5 million instructions per second depending on how the two busses are arbitrated. Intel states that with the bus arbitration chip, due out in 1983, that the number of processors and busses are unlimited in a crossbar bus configuration(Ref 18). Therefore the power is unlimited but at present time Intel only makes cages to hold 6 boards and at most 10 memory boards(Ref 41). Each memory board contains 256k bytes of ram(Ref 41).

BUS TRAFFIC

To economize the bus traffic of the Intel 432 the system bus is used differently. The system bus does not use the standard bus scheme, location followed by data pattern. Instead it has one address followed by a count of number of bytes and then the data(Ref 24,42). This setup is used for all packets, including processor to memory, memory to processor and processor to processor.

INPUT/OUTPUT

Input/output on the 432 is different from most microcomputers. The 432 does not allow interrupts to disturb the data processors. There exists a special processor chip, the interface processor(Ref 12,24), to handle all interrupts. The interface processor creates a window for an associated processor to look into the 432 memory and move data into and out of memory in a direct memory access type environment. The associated processor can be a smart terminal or another small processor, not a 432, principally an Intel 8086.

CHIP SET

The Intel 432 is a 3 piece chip set with a fourth and fifth chip due out in 1983(Ref 18). Mentioned earlier is the interface processor to handle the input/output. The other two chips are the instruction decoder chip and the execution unit chip. The instruction decoder chip fetches and decodes the instruction and informs the execution unit what to do with the data. The instruction decode chip and execution unit chip are on the same board. Two of these boards can be tied together to be used in fault tolerance checking. If the two outputs agree, the data out is allowed to continue, otherwise the fault line and fault light go on and the general data processor chips turn themselves off(Ref 24,42). The fourth and fifth chips are to be used in a crossbar bus framework. The fourth chip is a bus interface unit to do switching from one bus to another. The fifth chip is a memory interface unit to help in using the increased memory traffic.

USER'S MANUAL

The Intel 432 system is implemented using the VAX 11/780 under the VMS operating system located in room 245 of building 640, area B, Wright-Patterson Air Force Base, as its host. The host has on it the ADA compiler, the programs necessary to determine where the compiler errors, the linker to link the modules compiled and generate the object deck for the 432. There exist a link between the VAX and the Intel Series III. The link is used to move programs from the VAX to the Series III. The Series III is also connected to the iAPX 432/670. Intel has provided a program, PRIME, on the Series III which will load and start execution on the 432.

Use the VAX/VMS EDT screen oriented editor to create your program. It is advisable to keep separate the specification and the body due to the time stamp checker policy of the compiler (Ref 17).

Have a high level module exactly the same as attachment 1 with the only change being the name of the module in the pragma environment.

If you are using floating point numbers examine attachment 15, Trigonometric functions which uses attachment 14, Real operations. This is necessary since the failure to do so will result in the compiler failing during the object code generation.

When the program is ready to compile type in the following sentence when the prompt appears. All of the following commands used on the VAX are described in reference 14.

```
@[ACS]IDA file name (include the extension)
```

When the compiler returns with a statement of zero errors then type in the following command when the prompt appears. This example will assume that the file name used when compiling was FILE.NAM

```
@[ACS]REPLIST FILE.NAE FILE.NAL
```

Notice that the extension was changed from NAM to NAE. The E is used for environment files. The L is used for listings. The first two characters were not changed. The last character is described in reference 14.

When the compiler returns with a statement of non zero errors then type in the following command when the prompt appears.

```
@[ACS]REPORT file name (the last character of the
name is changed to R)
```

When all the programs have been compiled then it is time to link the modules. Examine [ACS]PRIME.LKD. Copy this file and change the name to what you desire. It is necessary to have the first module in the link statement be a previously linked module, this will normally be the operating system, iMAX. After the LKD file has been changed execute the following command when the prompt appears.

```
@[ACS]LINK432 file name (without the LKD ending)
```

Now assume that the 432 execuable object file you just create was named FILE.EOD. Print the object map, compiled code and compiled listing. This will be an extensive listing. You are now finished with the VAX.

On the Series III, execute the program ONLINE. This program will connect the Series III to the VAX as if it were a terminal. Log into the VAX and access the directory where FILE.EOD resides. Enter control C which will return the Series III to ISIS, the operating system. Then enter the following command.

```
DNLOAD FILE.EOD TO :F2:FILE.EOD
```

This assumes that the disk resides in drive 2 and that the disk has enough room. It is advisable to have an empty disk when down loading the file.

You are now ready to execute your program on the 432. Enter the following list of commands on the Series III.

```
RUN :F1:DEB432

INIT

INCLUDE :F1:DEB432.TEM

LOAD :F2:FILE.EOD

DEBUG

START
```

Your program is now executing on the 432. If you wish follow the directions found in the Reference 15 to help debug your program. Enter the command EXIT to leave the debug program.

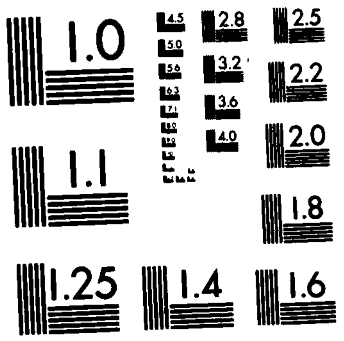
AD-A124 852 GENERATING AN OUT-THE-WINDOW COCKPIT IMAGE WITH THE IAPX 432(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING R A COOPER DEC 82 2/2

UNCLASSIFIED AFIT/GCS/EE/82-12 F/G 9/2 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SUMMARY

The Intel iAPX 432 micromainframe computer has the potential to be a widely used computer but there are many details to be worked out. The language ADA must be fully augmented, not just the current subset. Programmers must change from thinking in a serial sequence to thinking in a hierarchy frame. This machine should be good at doing multi-tasking. If the application does not warrant sub-tasking, then the Intel 432 is not a good choice because other machines can do the job faster and easier. The chip set is still undergoing changes and will not be finalized until early in 1983 (Ref 18). This computer has potential, but it first must be understood and that is not easy.

APPENDIX B

CONDUCTOR'S STATION

The operating system interface with the conductor's programs are through a high level program that sets up the systems initializations and executions. This program is found in attachment 1, CONDUCTOR. Figure 14 shows the structure of the conductor's program. From Figure 4 the program CONSITE was formed. Attachment 2 contains the program CONSITE. READ TEST SUBJECT bubble is the procedure call to READ_SUBJECT. Attachment 3 contains the package CONDUCTOR_WORK; READ_SUBJECT is found in CONDUCTOR_WORK. The DETERMINE SIGHTABILITY MATRIX bubble is the main loop with a call to I_SEE_U to create the matrix. Attachment 4 contains the package I_SEE_YOU which contains the procedure I_SEE_U. ASSEMBLE TEST SUBJECT'S PACKET bubble is accomplished in the procedure OUT_EQUAL_IN. OUT_EQUAL_IN is found in attachment 3. The packet is sent out with the procedure WRITE_SUBJECT. WRITE_SUBJECT is found in attachment 3.

Table II shows the correlation between the SADT charts from the requirements and the DFD diagrams of the design. Figures 4,5 and 7 are DFDs produced from the requirements. Figure 6 is an expansion of bubble number 2 in Figure 5.

Figure Number	DFD Diagram Number	SADT Node NUmber	Figure Number	Name
4	0	A0	1	The PACRAT System
5	1.0	A1	2	The Test Conductor
7	2.0	A2	3	The Test Subject

Table II Correlation between SADT and DFDs

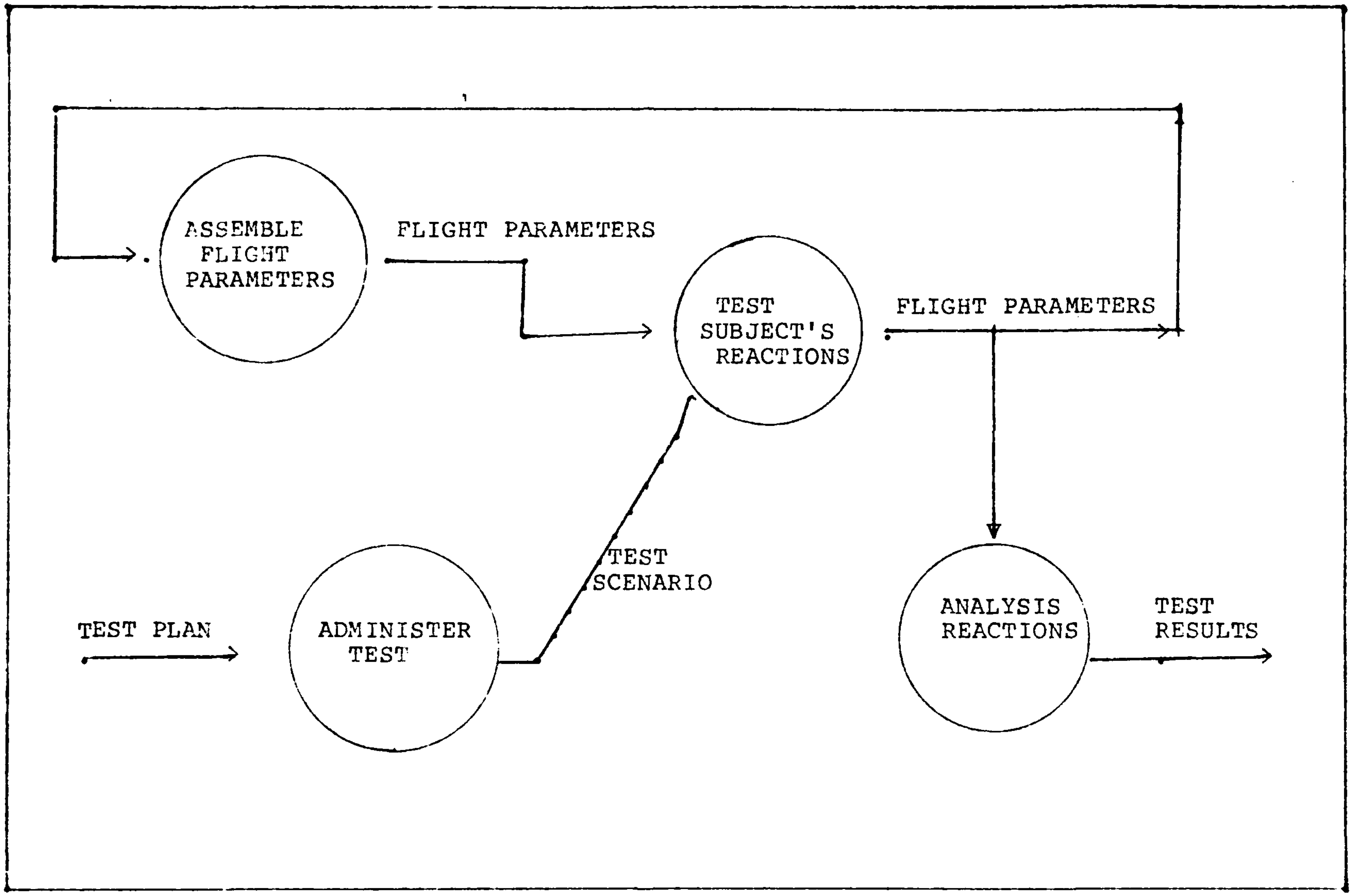


FIGURE 5 DIAGRAM 0 THE PACRAT SYSTEM

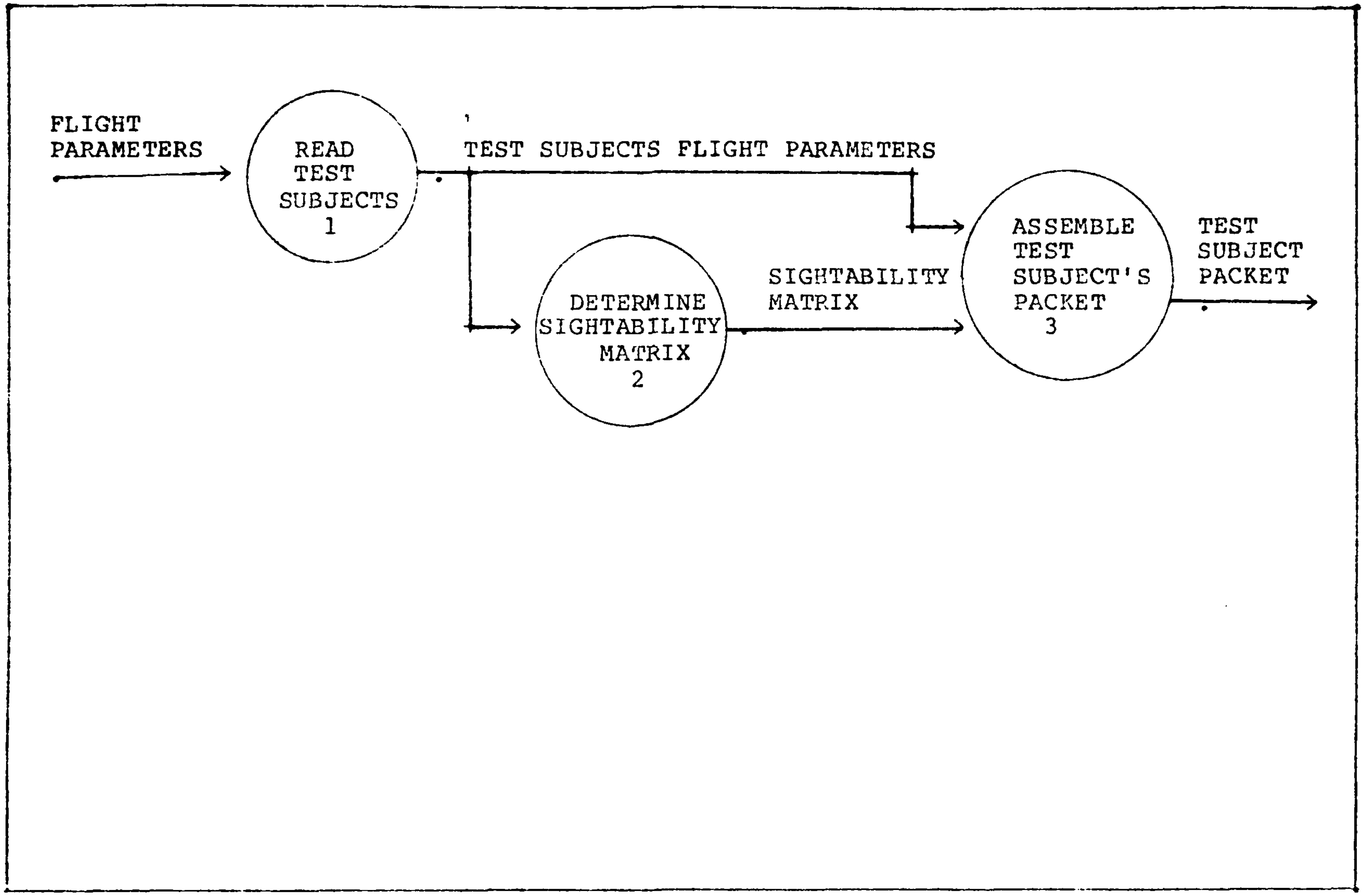

FLIGHT PARAMETERS
READ TEST SUBJECTS 1
TEST SUBJECTS FLIGHT PARAMETERS
DETERMINE SIGHTABILITY MATRIX 2
SIGHTABILITY MATRIX
ASSEMBLE TEST SUBJECT'S PACKET 3
TEST SUBJECT PACKET

FIGURE 6 DIAGRAM 1.0 PACRAT TEST CONDUCTOR

Table III show the correlation between the attachment number and the bubble found on a data flow diagram or a figure number. In Figure 14 is the structure chart of the procedures to be incorporated into the test conductor's initial station. Following that chart is Figure 15 with the structure according to packages.

ATTACHMENT NUMBER	FIGURE NUMBER	BUBBLE/FIGURE NAME	ATTACHMENT NAME
1			CONDUCTOR
2	5	TEST CONDUCTOR	CONSITE
3	5	READ TEST SUBJECT	CONDUCTOR_WORK
4	6	DETERMINE SIGHT	I_SEE_YOU

Table III Correlation between attachemnt and figure numbers

```
------------------------------------------------------------
|                                                          |
|       CONDUCTOR SIGHT CALCULATIONS                       |
|                                                          |
------------------------------------------------------------
    |          SUBJECTS |               | CONDUCTOR |
    |^          WITHIN ||^SIGHTABILITY ||^ DATA    ||
    ||SUBJECT    TEN  V|| MATRIX      V|| OUT     V|
    V  DATA     MILES  V                |           V
-------------  ----------------- SUBJECT||  -----------
| READ       | |               |  DATA  V|  |WRITE     |
|  SUBJECTS  | | I SEE YOU     |         |  | CONDUCTOR|
|            | |               |         |  |          |
-------------  -----------------         V  -----------
                                  ----------
                                  |IN      |
                                  |EQUALS  |
                                  |OUT     |
                                  ----------
```

FIGURE 14 Conductor's Functions Chart

```
-----------------------------------------------------------
|                                                         |
|       CONDUCTOR SIGHT CALCULATIONS                      |
|                                                         |
-----------------------------------------------------------
|PROCEDURE       |PACKAGE         |PACKAGE         |PACKAGE
|                |                |                |
|                +<---------------|---+----------->+
|                |                |   |            |
V                V                V   ^            V
---------------  -------------  ----------  ------------------
| CONDUCTOR   |  | CONDUCTOR |  | I      |  |                |
|  SIGHT CALC |  | WORK      |  | SEE    |  | REAL OPERATIONS|
|             |  |           |  | YOU    |  |                |
---------------  -------------  ----------  ------------------
|PROCEDURE       |CONTAINS        |PROCEDURE       |PROCEDURES
|                |PROCEDURES      |                |
|                |AND             |                |
V                |DATA            V                V
----------------             -------------  ----------------
| READ_SUBJECT |             |           |  |              |
| SQUARE_ROOT  |             | I_SEE_U   |  | ASIN         |
| I_SEE_U      |             |           |  | SQUARE_ROOT  |
| WRITE_       |             |           |  |              |
|   CONDUCTOR  |             |           |  |              |
----------------             -------------  ----------------
                                |PROCEDURE
                                |
                                V
                             -------------
                             |           |
                             | ASIN      |
                             |           |
                             -------------
```

FIGURE 15 Conductor's Packages Chart

CONDUCTOR uses five functions

1. CONDUCTOR SIGHT CALCULATION
2. CONDUCTOR WORK
3. I SEE YOU
4. REAL OPERATIONS (FOR SQUARE ROOT)
5. INTIO (FOR INTEGER IO)

DATA

PROCEDURE

1. CONDUCTOR SIGHT CALC -- Determines the distance from one test subject to all others

CONDUCTOR WORK IS

DATA

1. CONDUCTOR DATA IN
2. CONDUCTOR DATA OUT
3. PARAMETER

PROCEDURES

1. READ PARAMETER -- Reads PARAMETER
2. WRITE PARAMETER -- Writes PARAMETER
3. READ from the CONDUCTOR -- Reads CONDUCTOR DATA OUT
4. WRITE to the CONDUCTOR -- Writes CONDUCTOR DATA IN
5. READ from the SUBJECT -- Reads CONDUCTOR DATA IN
6. WRITE to the SUBJECT -- Write CONDUCTOR DATA OUT
7. OUT EQUAL IN -- Move conductor data in to conductor data out except for sightability array

THESE PROCEDURES USE INTIO PACKAGE FOR INTEGER IO

I SEE YOU IS

DATA

PROCEDURE

1. I SEE YOU -- Compute angle from airplane to other test subject this procedure uses real operations for the sine routine.

APPENDIX C

SUBJECT'S STATION

Subject's station is a program which executes at each subject station it starts by obtaining from the subject the station's id and aircraft type and starting conditions. Then while looping every 1/10th of a second it will read in the test conductor's output and determine which subjects can see other subjects and relay the aircraft description to the drawing program. Finally the drawing program will will take the new active data base and make a new drawing.

Below are data flow diagrams which show the design of the system. They are exact copies from the thesis text in chapter 3.

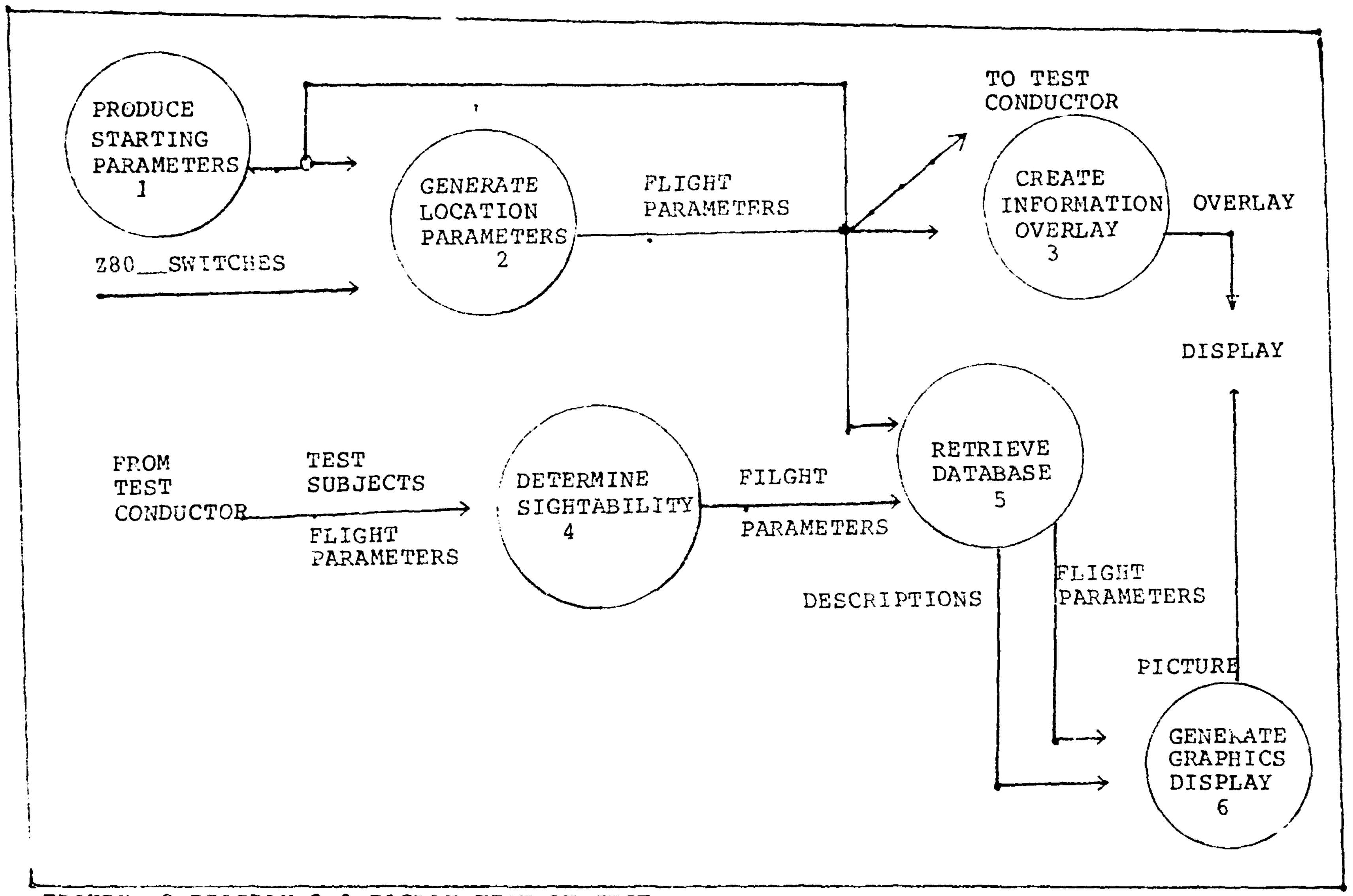


FIGURE 8 DIAGRAM 2.0 PACRAT TEST SUBJECT

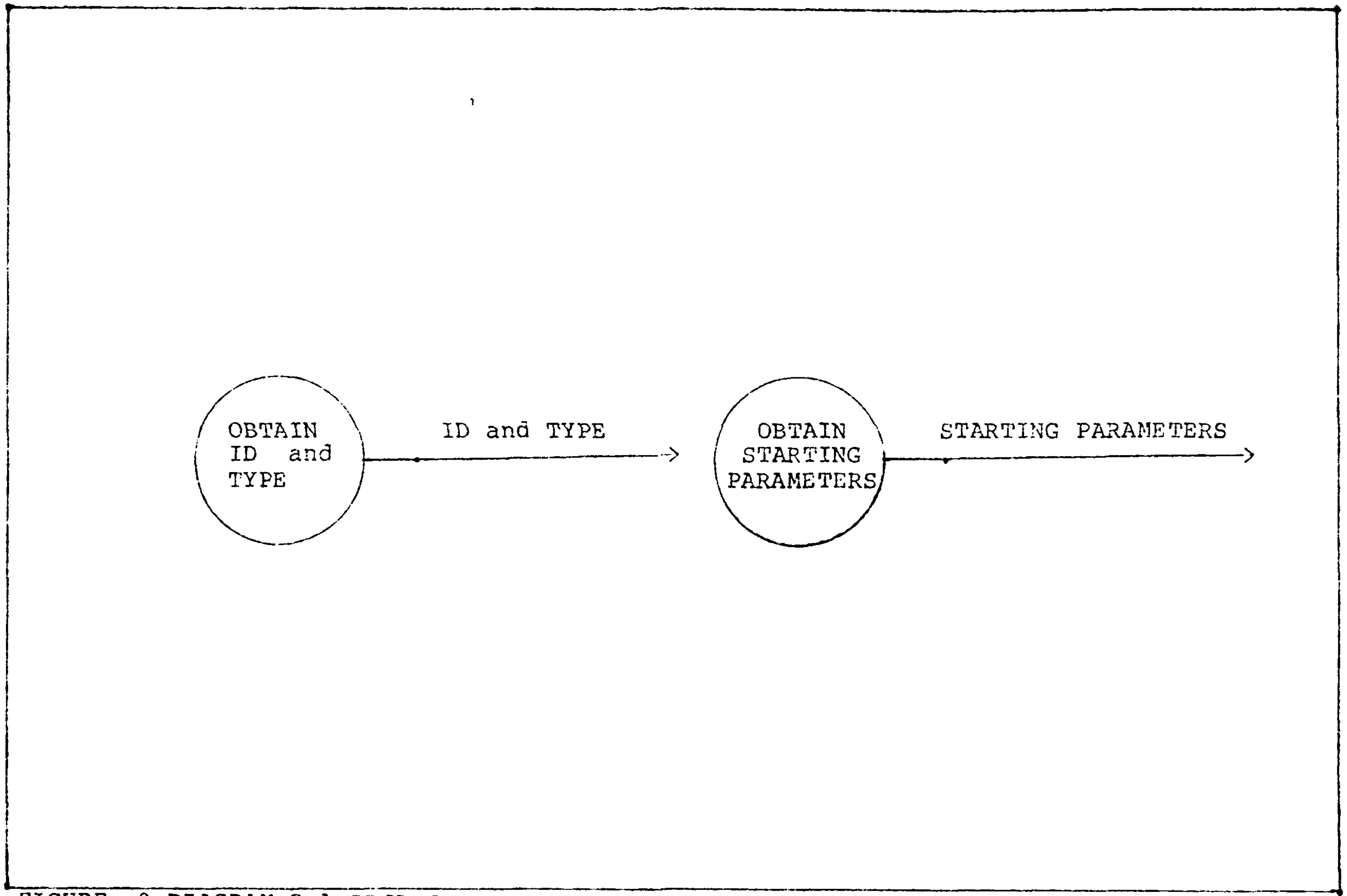


FIGURE 9 DIAGRAM 2.1 PRODUCE STARTING PARAMETERS

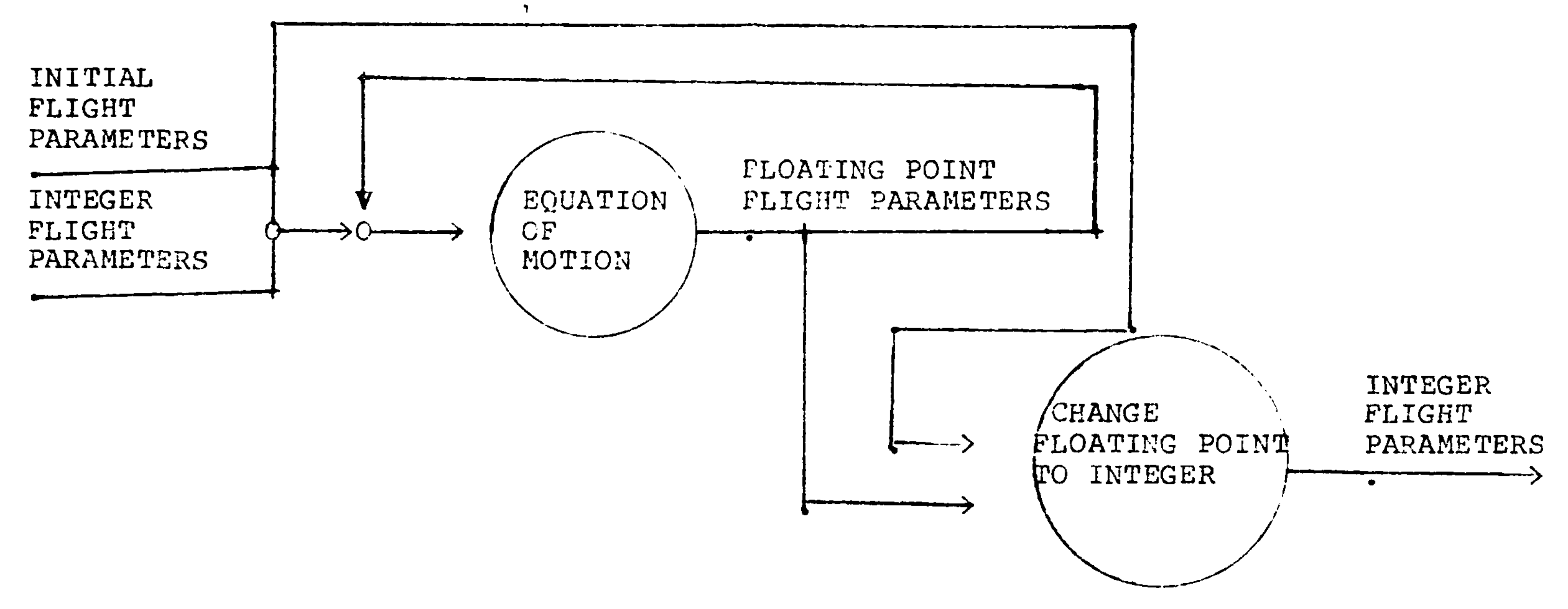

INITIAL FLIGHT PARAMETERS
INTEGER FLIGHT PARAMETERS
EQUATION OF MOTION
FLOATING POINT FLIGHT PARAMETERS
CHANGE FLOATING POINT TO INTEGER
INTEGER FLIGHT PARAMETERS

FIGURE 10 DIAGRAM 2.2 GENERATE LOCATION PARAMETER

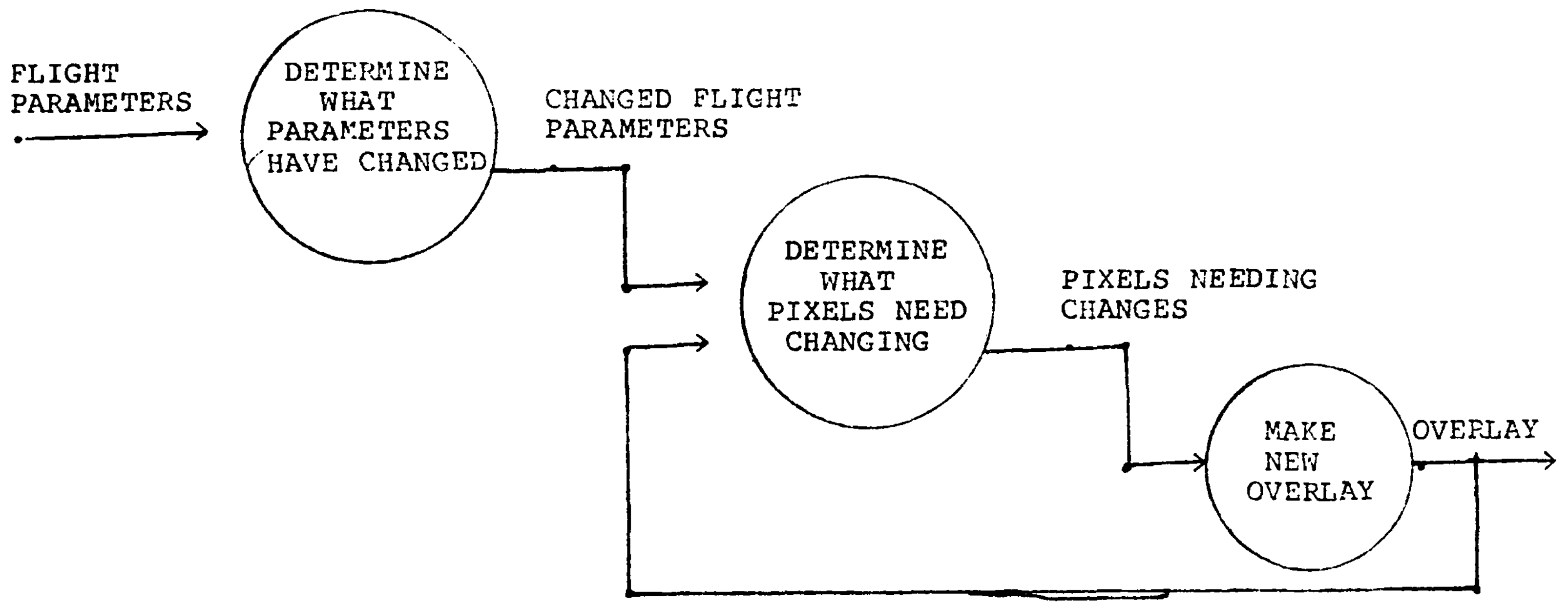

FLIGHT PARAMETERS
DETERMINE WHAT PARAMETERS HAVE CHANGED
CHANGED FLIGHT PARAMETERS
DETERMINE WHAT PIXELS NEED CHANGING
PIXELS NEEDING CHANGES
MAKE NEW OVERLAY
OVERLAY

FIGURE 11 DIAGRAM 2.3 CREATE INFORMATION OVERLAY

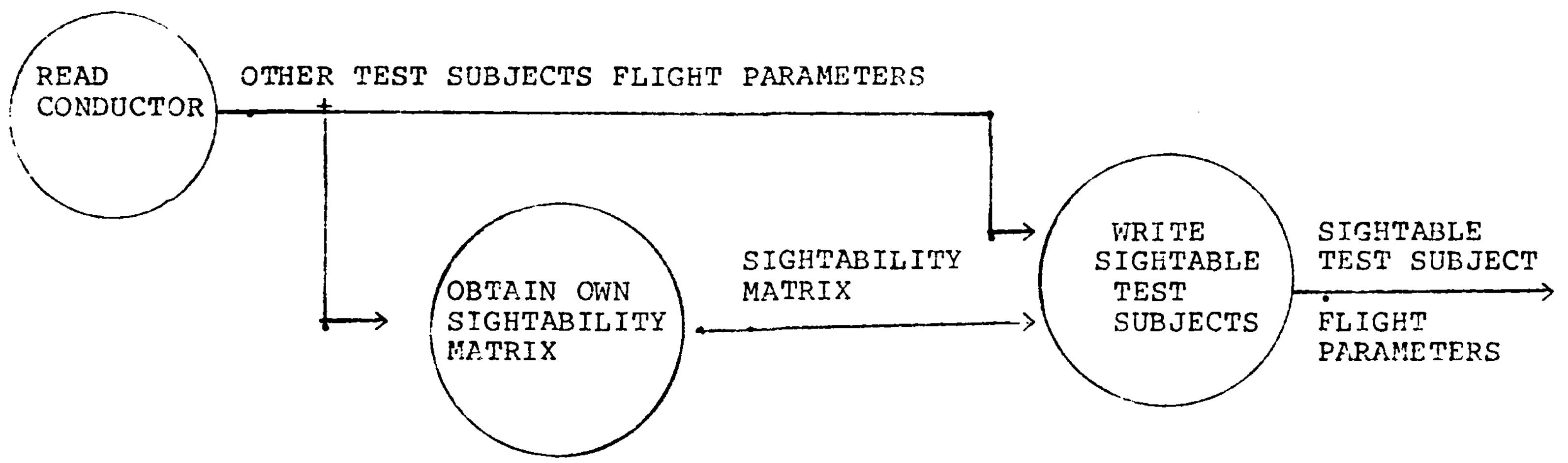

READ CONDUCTOR
OTHER TEST SUBJECTS FLIGHT PARAMETERS
OBTAIN OWN SIGHTABILITY MATRIX
SIGHTABILITY MATRIX
WRITE SIGHTABLE TEST SUBJECTS
SIGHTABLE TEST SUBJECT
FLIGHT PARAMETERS

FIGURE 12 DIAGRAM 2.4 DETERMINE SIGHTABILITY

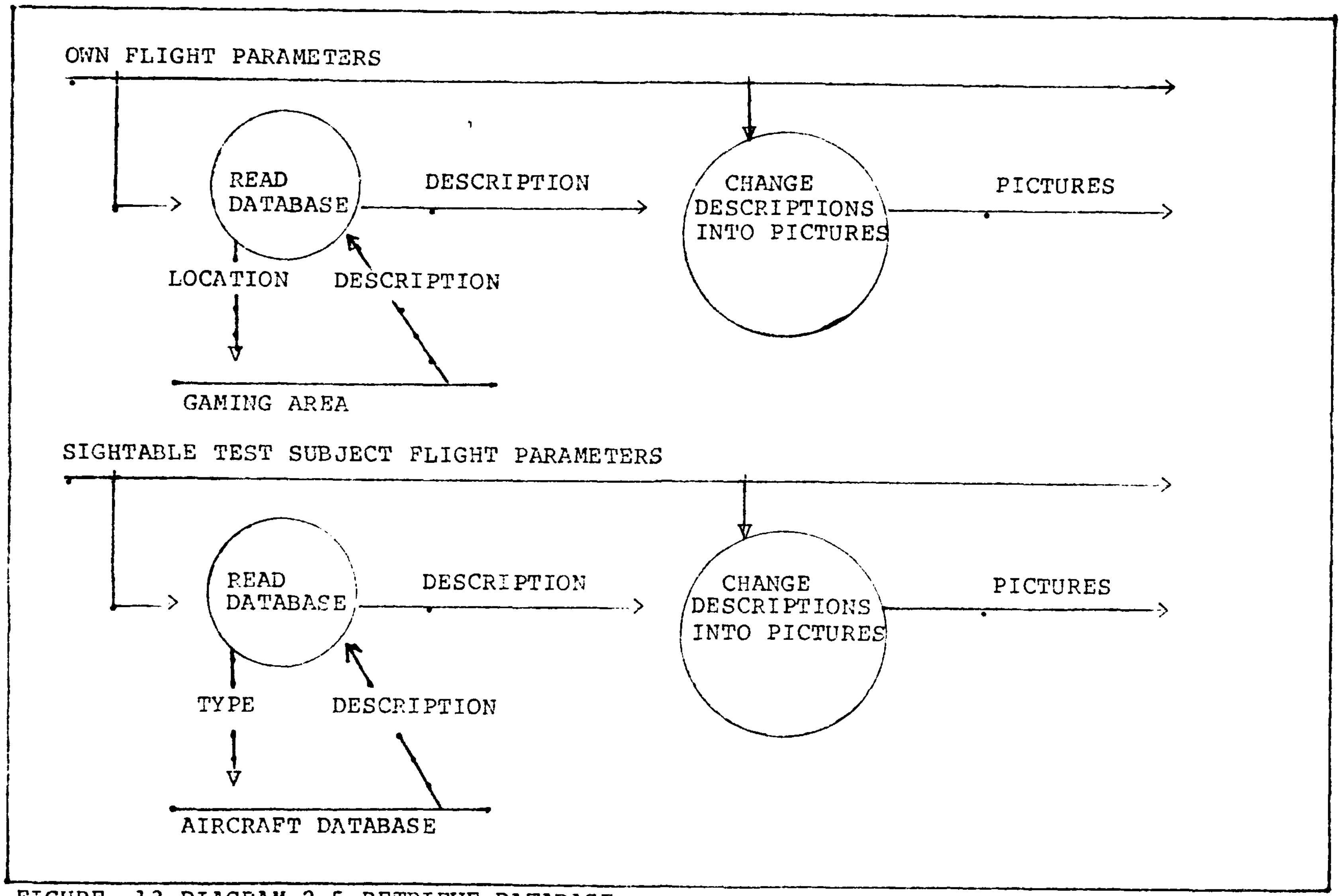


FIGURE 13 DIAGRAM 2.5 RETRIEVE DATABASE

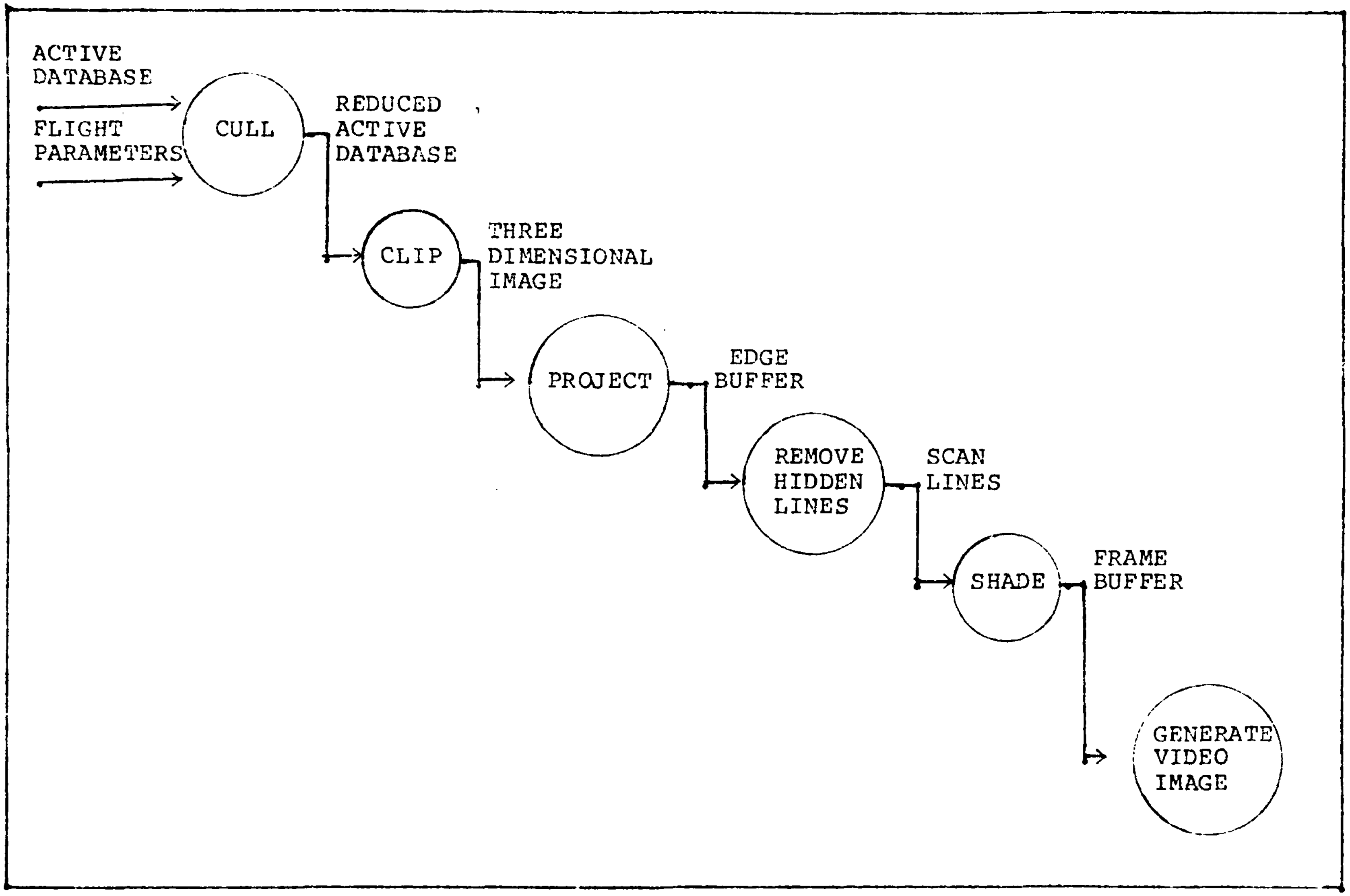


FIGURE 14 DIAGRAM 2.6 GENERATE GRAPHICS DISPLAY

The operating system interface with the subject's programs are through a high level program that sets up the systems initializations and executions. This program is found in attachment 5, TOP_OF_SUBJECT. From Figure 6 the program SUBJECT was formed. Attachment 6 contains the program SUBJECT. From Figure 7 the program found in attachment 7, INITIAL_TEST was coded. Program EQUATION_OF_FLIGHT contains the interface to the GENERATE LOCATION PARAMETERS, CREATE INFORMATION OVERLAY, and RETRIEVE DATA BASE procedures. This program is found in attachment 8. Program SIGHT_TO_DESCRIPTION contains the interface to the procedures DETERMINE SIGHTABILITY and RETRIEVE DATA BASE. This program is found in attachment 9. The bubble, GENERATE GRAPHICS DISPLAY, is found in package PICTURE, which is found in attachment 10.

EQUATION_OF_FLIGHT contains interface between three procedures. The GENERATE LOCATION PARAMETERS bubble is manifested in the package EQUATION_OF_MOTION, which is found in attachment 11. CREATE INFORMATION OVERLAY bubble is coded in the package INFORMATION_OVERLAY. INFORMATION_OVERLAY package is found in attachment 12. RETRIEVE DATA BASE bubble is manifested in package ACTIVE_DATABASE. EQUATION_OF_FLIGHT uses the procedure GET_AREA_DESCRIPTION in the package. The package is listed in attachment 13.

SIGHT_TO_DESCRIPTION contains the interface between two procedures. The DETERMINE SIGHTABILITY bubble is accomplished by use of the procedure READ_SUBJECT which is found in attachment 3, and the boolean variable set in procedure I_SEE_U, attachment 4. The RETRIEVE DATA BASE bubble is a call to the procedure GET_AIRCRAFT_DESCRIPTION found in attachment 13, ACTIVE_DATABASE.

Table IV summarizes the paragraphs above.

ATTACHMENT NUMBER	FIGURE NUMBER	BUBBLE/FIGURE NAME	ATTACHMENT NAME
5			TOP OF SUBJECT
6	7	TEST SUBJECT	SUBJECT
7	8	STARTING PARAMETER	INITIAL TEXT
8	9	GENERATE LOCATION	EQUATION FLIGHT
	10	INFORMATION OVERLAY	
	12	RETRIEVE DATA BASE	
9	11	DETERMINE SIGHTABLE	SIGHT DESCRIPTION
	12	RETRIEVE DATA BASE	
10	13	GENERATE DISPLAY	PICTURE
11	9	GENERATE LOCATION	EQOATION MOTION
12	10	INFORMATION OVERLAY	INFORMATION OVERLAY
13	12	RETRIEVE DATA BASE	ACTIVE DATA BASE

Table IV Correlation between figure and attachment numbers

```
---------------------------------------------------------------
|                                                             |
|                    SUBJECT                                  |
|                                                             |
---------------------------------------------------------------
|^INITIAL                    LOOPING
||PARAMETERS                    |
||ID AND              ----------+------------------
| TYPE                |CONCURRENT|                |
|                     |          |                |
|                     |          |                |
V                     V          V                V
-----------  ----------------  -------------  -----------
|INITIAL  |  |DETERMINE     |  |EQUATION   |  |MAKE     |
|TEXT     |  |SIGHTABILITY  |  |of FLIGHT  |  |PICTURES |
-----------  ----------------  -------------  -----------
```

FIGURE 16 Subject's Function Chart

```
------------------------------------------------------------
|                                                          |
|                  SUBJECT                                 |
|                                                          |
------------------------------------------------------------
|PROCEDURE   |2 PACKAGES|2 PACKAGES   |PACKAGE  |PACKAGE
|            |          |             |         |
|            +--------->+<---------+--|--------<|---+
|            |1 uses    |          ^  |         |   ^
|            |  --------+          |  |         |   |
V            |  |       V          ^  V         V   ^
------------ |  | ------------- ------------- -----------
|          | |  | | CONDUCTOR | |           | | MAKE    |
|  SUBJECT | |  | | WORK      | | EQUATION  | | PICTURE |
|          | |  | |           | | of FLIGHT | |         |
------------ |  | ------------- ------------- -----------
|PROCEDURE   |  |   |CONTAINS   |PROCEDURE    |PROCEDURES
|            |  |   |PROCEDURES |             |
|            |  |   |AND DATA   |             |
|            |  +-------+       |             |
|            +--+       |       |             |
V               |       V       V             V
--------------- | -------- --------------- ----------------
|INITIAL__TEXT| | |      | |READ SWITCH  | | CLIP         |
|SIGHT TO     | | |ACTIVE| |EQUATION     | | REMOVE       |
| DESCRIPTION | | |      | |  OF FLIGHT  | | SHADE        |
|EQUATION     | | |DATA  | |MAKE         | | COMBINE      |
|  OF FLIGHT  | | |      | |     OVERLAY | | WRITE_PICTURE|
|MAKE PICTURE | | |BASE  | |WRITE_PICTURE| |              |
--------------- | -------- --------------- ----------------
                |  |CONTAINS PROCEDURE
                |  |AND DATA
                |
+---------------+------------------------+
|                                        |
V                                        V
--------------------     --------------------------------
|                  |     |                              |
|INITIAL TEXT      |     | SIGHT TO DESCRIPTION         |
|                  |     |                              |
-------------------     --------------------------------
       |PROCEDURES                      |PROCEDURES
       |                                |
       V                                V
-------------------     --------------------------------
|                 |     |SIGHT TO DESCRIPTION          |
|INITIAL TEXT     |     |READ__GAME                    |
|INITIAL WRITE    |     |                              |
-------------------     --------------------------------
       |USES TEXT-IO            |USES CONDUCTOR WORK
                                |AND ACTIVE DATA BASE
```

FIGURE 17 Subject's Package chart

SUBJECT uses nine packages

1. CONDUCTOR WORK
2. ACTIVE DATA BASE
3. SIGHT TO DESCRIPTION
4. INITIAL TEXT
5. EQUATION OF FLIGHT
6. EQUATION OF MOTION
7. INFORMATION OVERALY
8. MAKE PICTURE
9. REAL OPERATIONS
10. TRIGONMETRIC FUNCTIONS

DATA

PROCEDURE

1. SUBJECT -- The main driver it will call the initialization routine then loops forever to make pictures.

1. CONDUCTOR WORK IS

DATA

1. CONDUCTOR DATA IN
2. CONDUCTOR DATA OUT
3. PARAMETER

PROCEDURES

1. READ PARAMETER -- reads PARAMETER
2. WRITE PARAMETER -- writes PARAMETER
3. READ CONDUCTOR -- reads CONDUCTOR DATA OUT

4. WRITE CONDUCTOR -- writes CONDUCTOR DATA IN

5. READ SUBJECT -- reads CONDUCTOR DATA IN

6. WRITE SUBJECT -- write CONDUCTOR DATA OUT

7. OUT EQUAL IN -- move CONDUCTOR DATA IN to
   CONDUCTOR DATA OUT except for
   SIGHTABILITY array

These procedures use INTIO package for INTEGER IO

2. ACTIVE DATA BASE IS

DATA

ACITVE_DATABASE_AIRCRAFT

ACTIVE_DATABASE_AREA

PROCEDURE

1. GET_AREA_DESCRIPTION -- Reads the gaming data base

2. GET_AIRCRAFT_DESCRIPTION
   -- Reads the aircraft data base

3. INITIAL TEXT IS

DATA-OUT

1. AIRCRAFT TYPE

2. AIRCRAFT ID

3. INITIAL PARAMETERS

PROCEDURE

1. INITIAL TEXT

2. INITIAL WRITE

4. SIGHT TO DESCRIPTION IS

USES

1. ACTIVE DATA BASE

2. CONDUCTOR WORK

DATA IN

1. AIRCRAFT ID

DATA OUT

1. ACTIVE DATA BASE AIRCRAFT

2. ACTIVE DATA BASE AIRCRAFT SIZE

PROCEDURES

1. SIGHT TO DESCRIPTION -- uses the conductor's work to determine what aircraft the subject can see, retrieves that data and passes it back to the main program.

5. EQUATION OF FLIGHT is

USES

1. ACTIVE DATA BASE

2. CONDUCTOR WORK -- FOR PARAMETER

3. INFORMATION OVERLAY

4. EQUATION OF MOTION

5. INTIO

6. TRIGONOMETRIC FUNCTIONS

7. REAL OPERATIONS

DATA IN

1. PARAMETER

DATA OUT

1. PARAMETER

2. ACTIVE DATA BASE AREA

3. ACTIVE DATA BASE AREA SIZE

PROCEDURES

1. FLIGHT -- Reads the inputs from the test subject and with the old aircraft parameters calls equation of motion to calulate new flight parameters then tasks out the writing the new parameters, then making of the information overlay and getting the area description.

6. EQUATION OF MOTION is

USES

1. REAL OPERATIONS
2. TRIGONOMETRIC FUNCTIONS
3. CONDUCTOR WORK

DATA IN

1. PARAMETER

DATA OUT

2. PARAMETER

PROCEDURE

1. EQUATION MOTION --Calulates new flight parameters

7. INFORMATION OVERLAY is

USES

1. CONDUCTOR WORK
2. PICTURE IMAGE
3. ACTIVE DATA BASE
4. INTIO

DATA IN

1. PARAMETER

DATA OUT

1. WRITES OUT AN OVERLAY

PROCEDURE

1. MAKE OVERLAY -- Takes the parameters just created and makes the information overlay for the graphichs image.

8. MAKE PICTURE is

uses

1. ACTIVE DATA BASE
2. PICTURE IMAGE
3. CONDUCTOR WORK

DATA IN

1. PARAMETER
2. ACTIVE DATA BASE
3. SIZE OF DATA BASE

DATA OUT

1. WRITES A PICTURE IMAGE

PROCEDURES

1. CLIP -- Determine what is in the viewing pryamid

2. REMOVE -- Remove hidden lines

3. SHADE -- Determine what color the pixels are from what surface it is on

4. COMBINE -- Takes all sections and combines them into one image.

5. MAKE GROUND PICTURE -- main routine to coordinate making ground picture.

6. MAKE AREA PICTURE -- main routine to coordinate making aircrafts picture.

9. REAL OPERATIONS is

USED

1. I432

DATA IN

1. REAL
2. INTEGER

DATA OUT

1. REAL
2. INTEGER

PROCEDURES

1. + -- adds two reals and returns a real
2. - -- subtracts two reals and returns a real
3. / -- divides two numbers, a real or an integer, and returns a real
4. * __ multiplies two reals and returns a real
5. ** -- takes a real raised to an integer time and return a real
6. SQRT -- takes a number and returns its square root

7. REALY -- takes a integer and returns a real

8. TEMPORARY_REALY -- takes a number and returns a temporary real

9. INTEGR -- takes a real and returns an integer

10. < -- compares two reals for less than and returns a boolean

11. > -- compares two reals for greater than and returns a boolean

12. & -- compares two reals for equal and returns a boolean

APPENDIX D

SYNOPSIS

BACKGROUND

Small computer processing power has been increasing since small computers first appeared on the market. However, generating solid three dimensional figures for a simulator requires a large amount of processing power (Ref 23:25). Historically simulators have used large mainframes or special purpose computers to handle the processing power required. If the requirements for the simulator were reduced to the minimal amount of throughput necessary to continue the human interface, then the processing power requirements would be reduced. Can a minicomputer meet the processing power requirements of a minimal simulator? The Biological Acoustics Branch of the Air Force Aerospace Medical Research Laboratory (AFAMRL/BBA) in researching audio communications require a minimal simulator to add a degree of authenticity to their test.

AFAMRL/BBA would like a simulator with solid three dimensional displays of a ground terrain and of the other test subjects airplanes that are in view. The PACRAT, Performance And Communication Research And Technology, system is the test which requires the use of the simulator. PACRAT is a system by which subjects' audio communication skills are tested. The main skill tested is hearing with interference from outside sources. The current test includes trying to maintain a crosshair inside a moving circle on a video display while the audio communications are going on. The simulator will replace the crosshair in the moving circle when the simulator is ready. The test results include how often the subject kept the crosshair inside the circle and how frequently the test subject understood the commands given by the test conductor over the audio communication lines. Reference 23 did a preliminary requirements review of the PACRAT system and Chapter 2 of this thesis will summarize the requirements.

After much studying and compiling of facts (Ref 23) it became evident that a digitized data base is required to produce the ground terrain picture. Reference 23 described the available data bases and concluded that the F111 DIG digitized ground terrain data base was the best suited to the PACRAT system needs. This data base has a 32 bit wide key to access the data base. The 32 bit wide key of the data base produces a requirement for a 32 bit computer. There are several new 32 bit small computers available on the market today. After reviewing the literature on these new computers Reference 23 concluded that the Intel iAPX 432 might meet the processing power required to provide a minimal simulator.

SCOPE

This thesis investigation will study the use of the Intel iAPX 432 Micromainframe, its object oriented architecture, and the ADA programming language to support the PACRAT system. ADA is the only programming language which compiles to run on the 432. Programs necessary for the PACRAT system will be developed through top down design techniques. The development will consist of the following:

-- Requirements review

-- Produce high level design of the system

-- Produce low level functional design of the modules in the system

-- Implement the design in ADA

-- Test and integrate the system

The final area of study will consist of examining the F111 DIG data base. This data base is needed to produce the ground terrain picture. This data base is currently on a tape with a Singer-Link operating system file format. This part of the thesis effort will read the tape and convert the internal machine format data into human readable format and then into a useable data base for the PACRAT system.

CURRENT KNOWLEDGE

This thesis investigation will continue on from where Reference 23 ended. The following was obtained from Reference 23.

- -Requirements as described by the sponsor which will be summarized in Chapter 2.
- -SADT charts for displaying the requirements which will be used in Chapter 2.
- -Hidden line algorithm for drawing screen images as seen by the author which will be used in Chapter 4.

The sponsor has a completed FORTRAN program for calculating the new position of a C135 cargo aircraft.

The following are still needed to implement the system.

-A ground terrain data base.

-A program design of the system flow.

-Timing studies of the iAPX 432 performing the task.

-A system integrating all the parts.

APPROACH

There will be three major efforts in this thesis aimed at implementing the PACRAT system. The first effort will be to try to acquire the F-111 visual data base for PACRAT. After acquiring the data base then work the data base into a useable data base for PACRAT. The second effort will be to analyze the Intel iAPX 432 architecture in relation to the PACRAT requirements. ADA will be studied for use in the 432 environment. The third major effort of this thesis investigation will analyze the requirements set forth (Ref 23:118) and produce an implementation for testing. The implementation is to be a design of an Out-The-Window Cockpit Image generator. The design will deal with reviewing the requirements already established (Ref 23) and the sponsor's new requirements (Ref 26). The Structured Analysis and Design Technique, SADT (Ref 39), approach will be studied and transformed into Data Flow Diagrams, DFDs (Ref 39), to show how requirements transform into design. Structure charts (Ref 39) will be presented to display the heirarchy of the packages and procedures in the system. Then the implementation of the programs will be achieved. The flow of data between modules, how the multi-tasking will interface, and the timing requirements will be presented. Finally the results of the design work and the timing results of the program on the Intel iAPX 432 will be compared with the requirements stated in Chapter 2.

RESULTS

Below are the high level SADT charts. The node A-0 represents the overall system. The A0 node describes the PACRAT test. The nodes A1 and A2 show the requirements of the test conductor and test subjects respectively.

NODE	TITLE	FIGURE
A-0	PACRAT System	1
A0	Test System	2
A1	Test Conductor	3
A2	Test Subject	4

Table I Correlation between Nodes and Figures

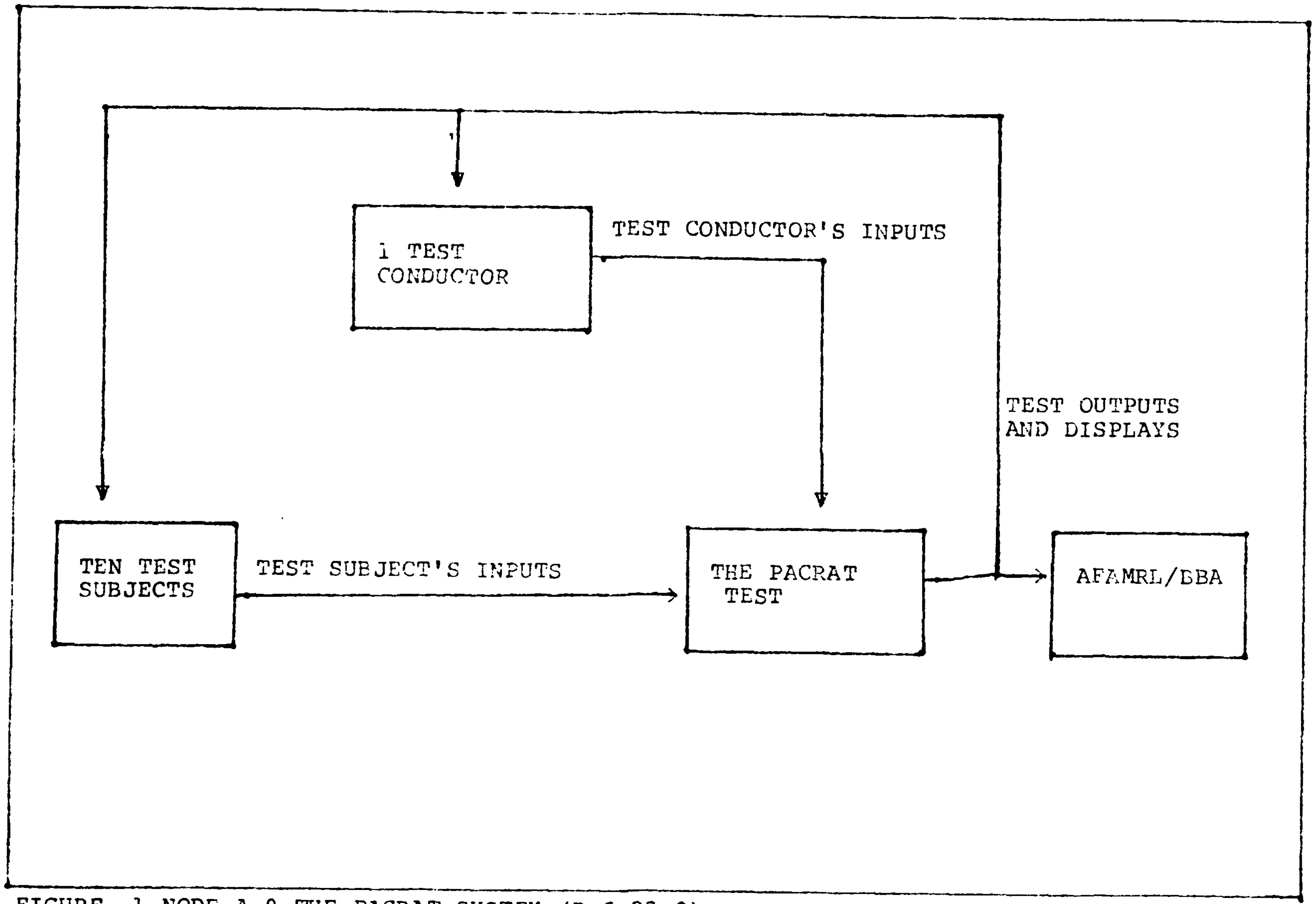


FIGURE 1 NODE A 0 THE PACRAT SYSTEM (Ref 23:9)

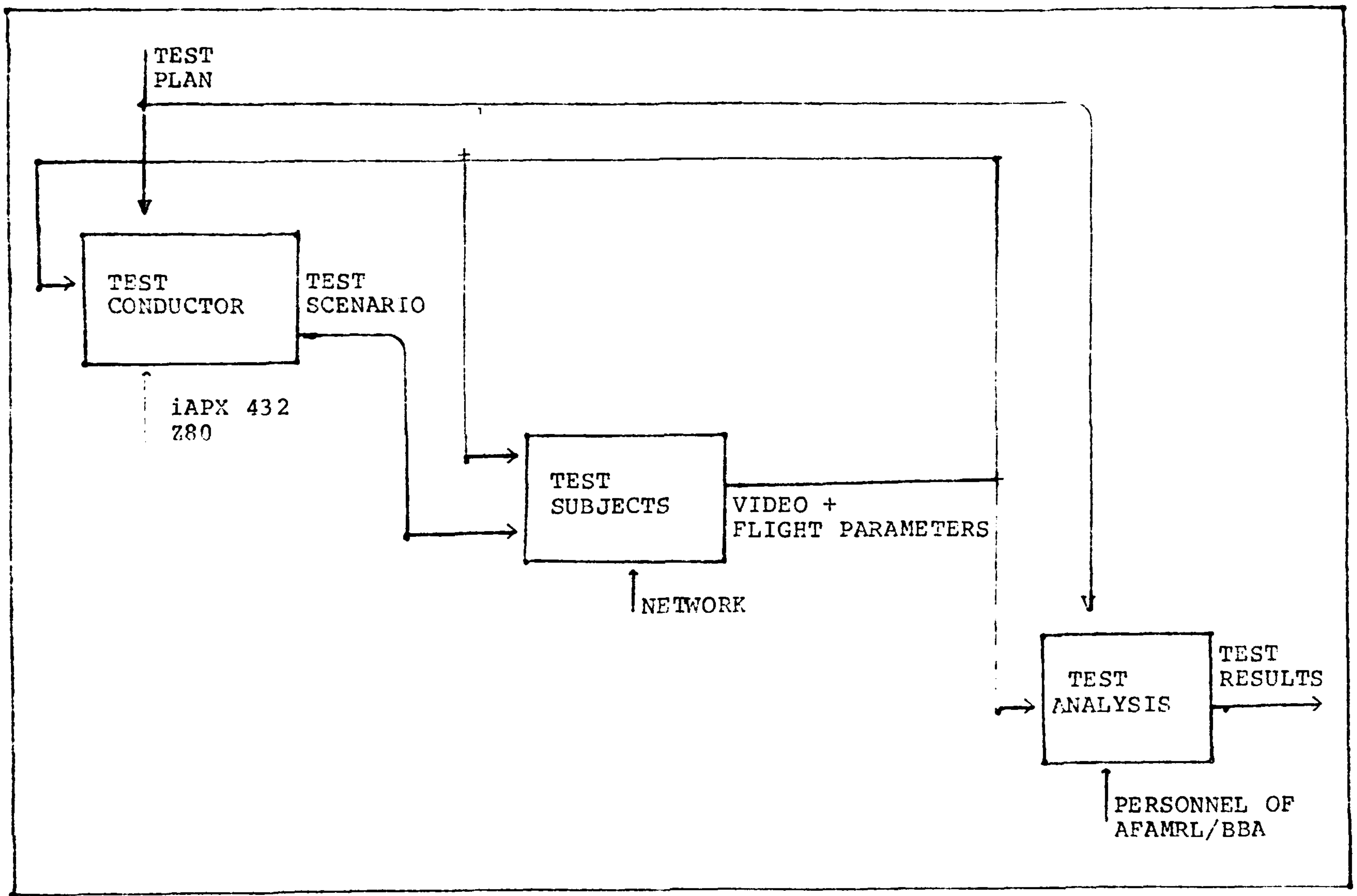


FIGURE 2 NODE A0 THE PACRAT TEST (Ref 23:11)

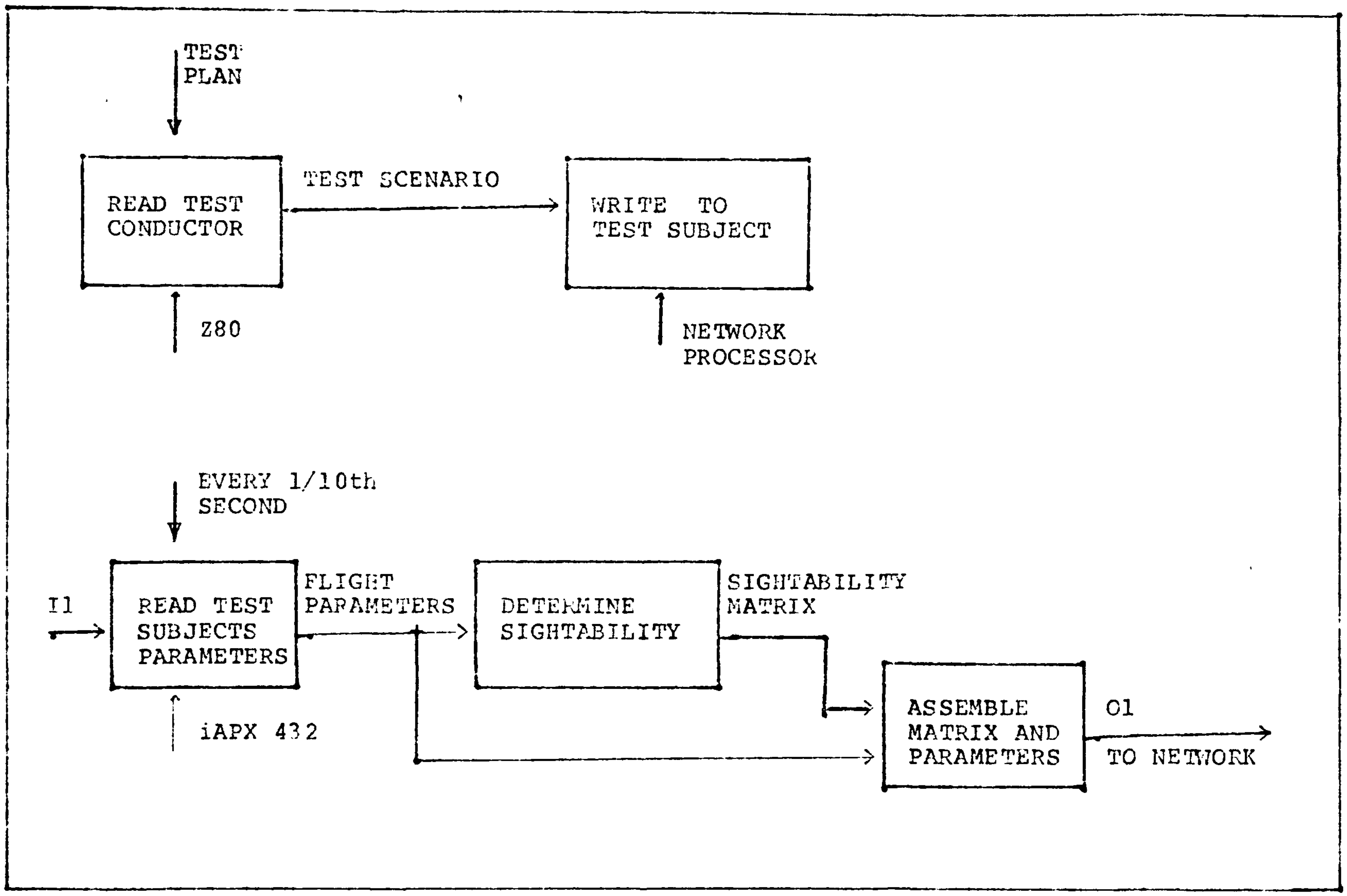


FIGURE 3 NODE A1 TEST CONDUCTOR

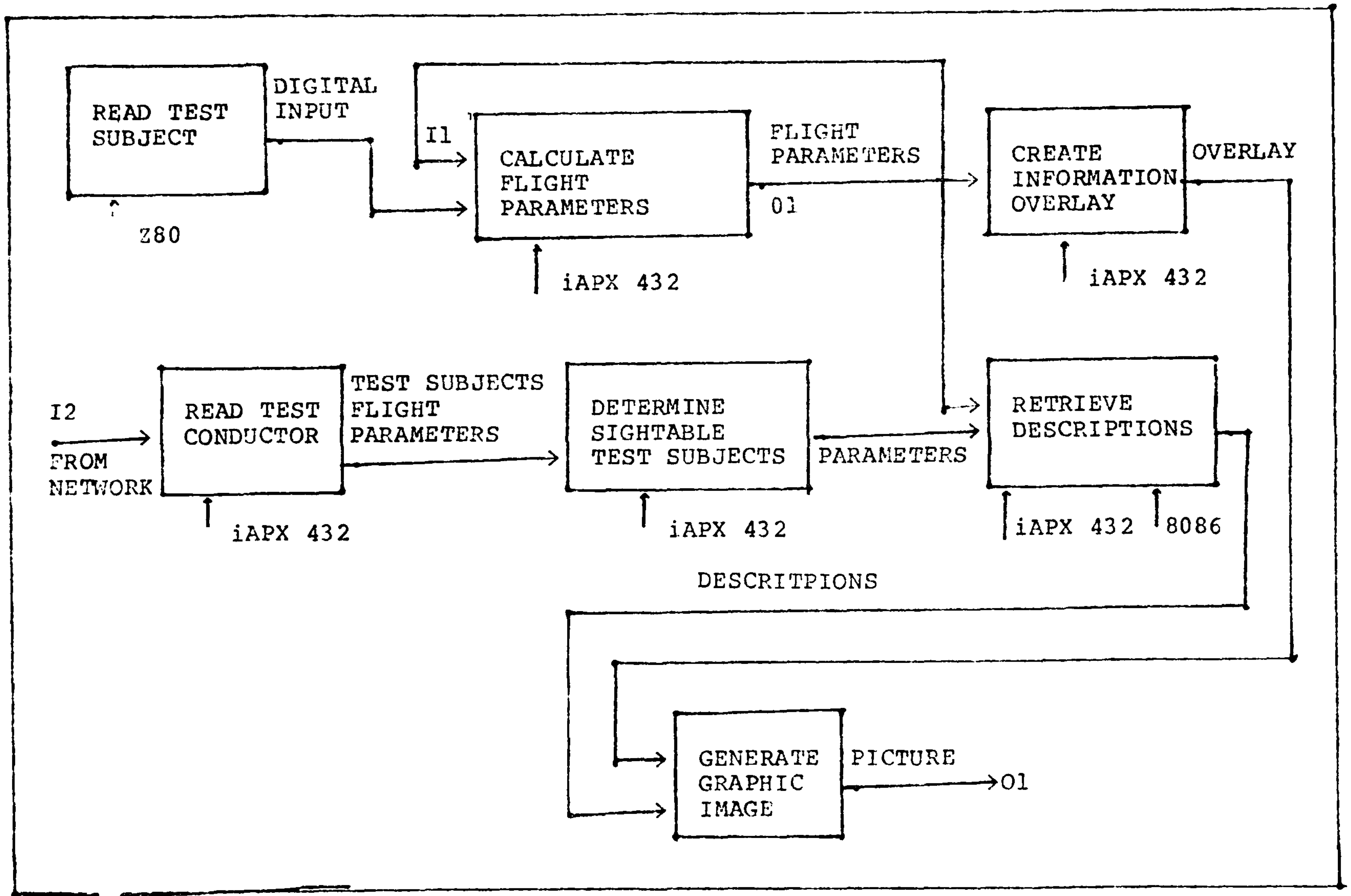


FIGURE 4 NODE A2 TEST SUBJECT

The results of this thesis investigation were less than optimal. The Intel 432 was looked at through the eyes of Intel's authors and not many others. The 432 did not become operational until after this thesis effort was completed. The results of this work include the requirements, design and implementation in ADA of a reduced simulator for the PACRAT system. The results should be varified when the 432 becomes operational and the new releases of the supporting software for the 432 arrives.

VITA

Roger A. Cooper was born on 28 April 1955 in Piqua, Ohio. He graduated from Lehman High School in Sidney, Ohio, in 1973 and then attended Ohio State University from which he received a Bachelor of Science degree in Mathetical Science applied in Industrial Engineering in June, 1977. Upon graduation, he received a commission in the United States Air Force through the Reserve Officers Training Corps program. His first assignment in the Air Force was as a Programmer/Analysis at Headquarters Strategic Air Command , Offutt Air Force Base, Omaha, Nebraska 68113. In June, 1981, he entered the School of Engineering at the Air Force Institute of technology. He is married to the former Vicki L. Berry and has two daughters, Jennifer and Patricia. His follow on assignment is to 4501 Computer Support Squadron, Langley Air Force Base, Newport News, Virginia.

Permanent Address: 724 Robinson Ave.
Piqua, Ohio 45356

END

FILMED

3-83

DTIC